
175 PTAS
mi computer
70
CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR
PENTAX
Nikon
EPSON
Editorial Delta, S.A.

MI COMPUTER CURSO PRACTICO
DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona

Volumen VI-Fascículo 70

Director:	José Mas Godayol
Director editorial:	Gerardo Romero
Jefe de redacción:	Pablo Parra
Coordinación editorial:	Jaime Mardones
	Francisco Martín
Asesor técnico:	Ramón Cervelló

Redactores y colaboradores: G. Jefferson, R. Ford, F. Martín, S. Tarditti, A. Cuevas, F. Blasco
Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Paseo de Gracia, 88, 5.°, 08008 Barcelona
Tels. (93) 215 10 32 / (93) 215 10 50 - Télex 97848 EDLTE

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador*, se publica en forma de 96 fascículos de aparición semanal, encuadernables en ocho volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-7598-034-7 (tomo 6)
84-85822-82-X (obra completa)
Depósito Legal: B. 52/1984

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 158505

Impreso en España-Printed in Spain-Abril 1985

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, 28034 Madrid.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Edificio Bloque Dearmas, final Avda. San Martín con final Avda. La Paz, Caracas 1010.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (96 fascículos más las tapas, guardas y transferibles para la confección de los 8 volúmenes) son las siguientes:

a) Un pago único anticipado de 19 425 ptas. o bien 8 pagos trimestrales anticipados y consecutivos de 2 429 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 6.850.277 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Editorial Delta, S.A. (Paseo de Gracia, 88, 5.°, 08008 Barcelona), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Editorial Delta, S.A., en la forma establecida en el apartado b).

Para cualquier aclaración, telefonear al (93) 215 75 21.

No se efectúan envíos contra reembolso.



Intercambio telefónico

Iniciamos una serie en que analizaremos en detalle las aplicaciones prácticas de las comunicaciones mediante modems

Tono de instrucción
Desarrollado originalmente para conectar terminales remotos a ordenadores centrales, el modem ("*mo*dulador/*dem*odulador") convierte, o modula, datos electrónicos digitales en tonos de audio para transmisiones telefónicas, y demodula los tonos de audio para volver a convertirlos en señales digitales en el momento de su recepción. Los acopladores acústicos transmiten y reciben a través del tubo del teléfono, mientras que los modems "compactos" se conectan directamente a la línea, por lo general a través de un conector de ampliación

La conexión a un modem le permite a su micro "hablar" con otros ordenadores a través del teléfono. De este modo, se abre ante usted una amplia gama de actividades relacionadas con la comunicación: enviar cartas que se reciben en el instante en que se transmiten, intercambiar software con amigos que vivan en lugares alejados, pasarse mensajes con otros usuarios o ganar acceso a un ordenador central. En un futuro artículo analizaremos estas aplicaciones. Aquí vamos a echar una mirada a los principios en los cuales se sustenta la transmisión de datos entre ordenadores.

La tecnología de las comunicaciones (también conocida como *coms*) tiene sus orígenes en la informática de ordenadores centrales. En el pasado se solía situar el ordenador central en una habitación construida expresamente y dotada de aire acondicionado, y desde allí se efectuaban conexiones con terminales distribuidos por todo el edificio. Un terminal constaba tan sólo de una pantalla y un teclado conectados al ordenador principal a través de un cable en serie. De este modo un usuario que estaba sentado en un extremo del edificio podía acceder al ordenador del otro extremo.

El enlace en serie directo entre el ordenador y el terminal funciona correctamente entre distancias relativamente pequeñas, es decir, de unos pocos centenares de metros; pero la pérdida de definición de las señales hace imposible la transmisión a través de distancias mayores, aun cuando el costo del cableado no represente obstáculo alguno. Éste es el motivo por el cual se desarrolló el modem.

Un *modem* (de "*mo*dulador/*dem*odulador") es un dispositivo que permite que los datos del ordenador se transmitan a través de una línea telefónica normal. Funciona convirtiendo las señales eléctricas del ordenador en tonos de audio de frecuencia y volumen aptos para la transmisión a través de la

red telefónica. Este proceso es lo que se conoce como *modulación*. El modem del ordenador receptor vuelve a convertir estos tonos de audio en señales eléctricas que se le puedan pasar al ordenador receptor (lo que se denomina *demodulación*). Se utiliza una señal constante (llamada *tono de portadora*) a modo de referencia, mientras se transmiten los datos en la onda modulada.

El resultado concreto de este proceso es que se puede acceder a ordenadores remotos casi como si estuvieran conectados directamente al terminal. Los terminales exclusivos por lo general están formados por una VDU y un teclado y, puesto que no poseen capacidad de proceso propia, se les suele calificar como "terminales tontos". Se puede emplear un microordenador como si fuera un terminal, simplemente utilizando un software de comunicaciones apropiado. Sin embargo, dado que un micro sí posee capacidad de proceso propia, se dice que en este caso es un "terminal inteligente".

Tipos de modem

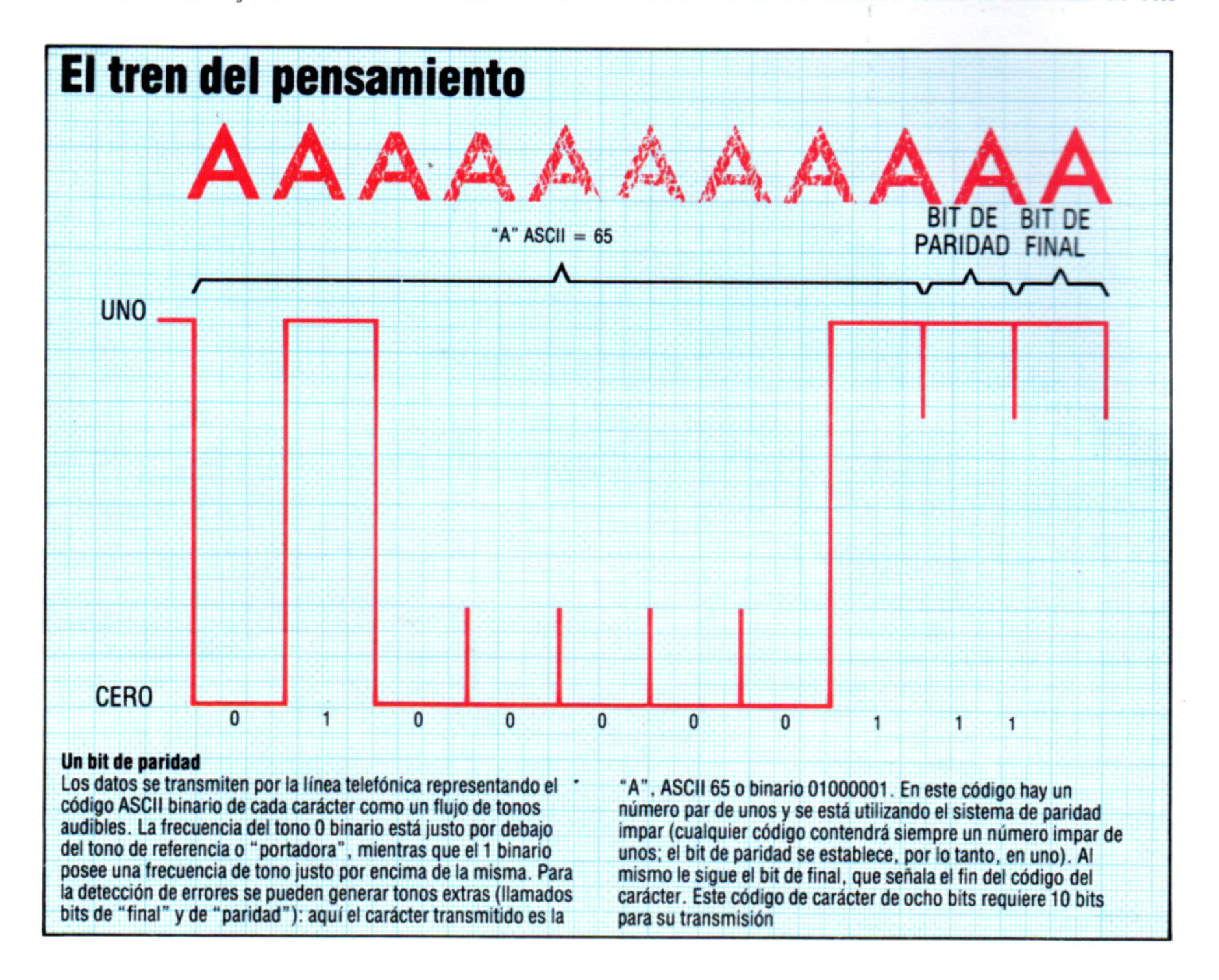
Existen dos tipos de modems: acústicos y compactos. Los *modems acústicos* (llamados por lo general *acopladores acústicos*) poseen dos tazas de goma en las cuales se coloca el tubo del teléfono. Los sonidos de audio se transmiten a través de la bocina y se reciben por el auricular. Los modems acústicos son los más convenientes de los dos tipos, porque se los puede utilizar con cualquier teléfono, ¡y los que funcionan a pilas se pueden incluso emplear con ordenadores portátiles para efectuar llamadas desde teléfonos públicos! Por otra parte, los modems acústicos están sujetos a la interferencia del ruido circundante y, por lo tanto, no son muy fiables.

Los *modems compactos* o de "conexión directa" se enchufan directamente en un enchufe estándar de teléfono y el teléfono se conecta en la unidad del modem. Además de ser más fiables que sus equivalentes acústicos, los modems compactos por lo general ofrecen más características.

Debido a que los modems compactos se enchufan directamente en la red telefónica, éstos han de haber sido homologados por la Compañía Telefónica Nacional de España (C.T.N.E.); esto asegura que no exista ninguna posibilidad de que se permita el paso de voltajes de la red eléctrica al sistema telefónico, y verifica que el modem se desconecte "limpiamente" al final de una llamada y no deje la línea "ocupada". El empleo de un modem no homologado es ilegal.

Entre las útiles características adicionales que ofrecen algunos modems se incluyen las de "respuesta automática" y "llamada automática" (esta última, todavía no implementada por la C.T.N.E.). Los modems con respuesta automática, al contestar al teléfono y detectar otro modem al otro extremo del hilo, le pasan el control al ordenador. Si la llamada no es de otro modem, el modem con respuesta automática simplemente cuelga. Los modems con llamada automática pueden aceptar un número desde el ordenador y "marcarlo" automáticamente. Por lo tanto, en respuesta a un nombre que entre el usuario, el software del ordenador puede buscar el número de teléfono en una base de datos y después indicarle al modem que marque ese número.

El *baudio* es la unidad de medida de la velocidad de transmisión de datos entre dos dispositivos. Normalmente se la considera como la cantidad de bits



Un bit de paridad
Los datos se transmiten por la línea telefónica representando el código ASCII binario de cada carácter como un flujo de tonos audibles. La frecuencia del tono 0 binario está justo por debajo del tono de referencia o "portadora", mientras que el 1 binario posee una frecuencia de tono justo por encima de la misma. Para la detección de errores se pueden generar tonos extras (llamados bits de "final" y de "paridad"): aquí el carácter transmitido es la "A", ASCII 65 o binario 01000001. En este código hay un número par de unos y se está utilizando el sistema de paridad impar (cualquier código contendrá siempre un número impar de unos; el bit de paridad se establece, por lo tanto, en uno). Al mismo le sigue el bit de final, que señala el fin del código del carácter. Este código de carácter de ocho bits requiere 10 bits para su transmisión



que se transmiten por segundo, si bien en la práctica ésta no es una definición exacta, por razones que veremos más adelante en este mismo capítulo.

Las dos velocidades más comunes para los dispositivos de comunicaciones son 300 y 1 200/75 (significando esta última que los datos se reciben a 1 200 baudios y se transmiten a 75). La mayoría de los micros, a título comparativo, guardan los programas en cinta a una velocidad similar (entre 300 y 1 200 baudios).

La velocidad de 300 baudios se utiliza para sistemas en los que se transmite aproximadamente la misma cantidad de datos en ambos sentidos. Éstos incluyen los boletines y los sistemas de correo electrónico como el Telecom Gold. La velocidad de 1 200/75 baudios se emplea para sistemas de videotexto como el Prestel, en los cuales la mayor parte de la información se envía en una sola dirección.

Lamentablemente, tal como ocurre con otros muchos aspectos de la informática, estos estándares no son universales. En razón de las diferencias existentes entre los sistemas telefónicos británico y norteamericano, por ejemplo, los dos países utilizan frecuencias distintas. La frecuencia de Gran Bretaña se conoce como el estándar CCITT (o V21), mientras que la de Estados Unidos es el tono Bell (en honor de la empresa telefónica).

Todo sobre bits

Debido a que el sistema telefónico sólo puede diferenciar con toda fiabilidad una cantidad limitada de frecuencias, los datos se transmiten en forma binaria. Para hacer esto, cada carácter se traduce primero a su equivalente ASCII y luego a su forma binaria. Por consiguiente, la letra "A" se convertiría en 65 (ASCII) y luego en 01000001 (binario). Los unos se representan mediante una frecuencia (justo por encima del tono de portadora y, por ello, se la conoce como "alta"), y los ceros mediante otra (justo por debajo del tono de portadora y, por tanto, considerada "baja").

Dado que el ordenador receptor (conocido como "anfitrión" o *host*) necesita alguna forma de saber dónde termina un carácter y comienza el siguiente, también se utilizan bits de comienzo y de final. Éstos son simplemente frecuencias convenidas; cuando el anfitrión recibe un bit de final, por ejemplo, decodifica los bits precedentes. El protocolo más común para las comunicaciones ASCII es de ocho bits de datos seguidos de un bit de final. Una posible alternativa es la de siete bits de datos y dos bits de final. Otros sistemas utilizan tanto bits de comienzo como de final, indicando el cambio el final de un carácter.

Puesto que la comunicación a través de la red de teléfonos pública no es absolutamente fiable, necesitamos algún método de verificación de errores, gracias al cual el anfitrión pueda asegurarse de que ha recibido un carácter correctamente. La solución más simple es lo que se conoce como "control de paridad". Ésta implica contar la cantidad de bits altos impares (sistema de *paridad impar*) o bien pares (sistema de *paridad par*). Por ejemplo, retomando el ejemplo de la letra "A", el número total de bits altos de 01000001 es dos (un número par). Por consiguiente, de acuerdo a la paridad impar, el bit de paridad también tendrá que estar alto. En el caso de la letra "C", sin embargo, que se traduce como 01000011 en binario, el bit de paridad tendría que estar bajo.

El control de paridad es relativamente poco sofisticado; detectará un error de un único bit, pero no detecta dos errores en el mismo carácter, porque entonces la paridad será correcta. No obstante, es simple y útil para la mayoría de las aplicaciones en las que no resulte vital una exactitud estricta.

Hemos mencionado que la definición de bits por segundo no es exacta para la definición de baudio. El motivo de ello es que se han de tener en cuenta bits de comienzo, de final y de paridad. Por ejemplo, en un sistema con control de paridad que utilice ocho bits de datos y un bit de final para cada carácter, sólo 8 bits de cada 10 transmitidos contienen información útil. Por lo tanto, los verdaderos bits de datos por segundo son el 80 % de 300 baudios, es decir, 240 baudios.

Hemos hecho un análisis detallado de los principios fundamentales de la transmisión de datos mediante modems. En el próximo capítulo consideraremos otros aspectos del funcionamiento del modem, como transmisión dúplex y protocolos de terminales.



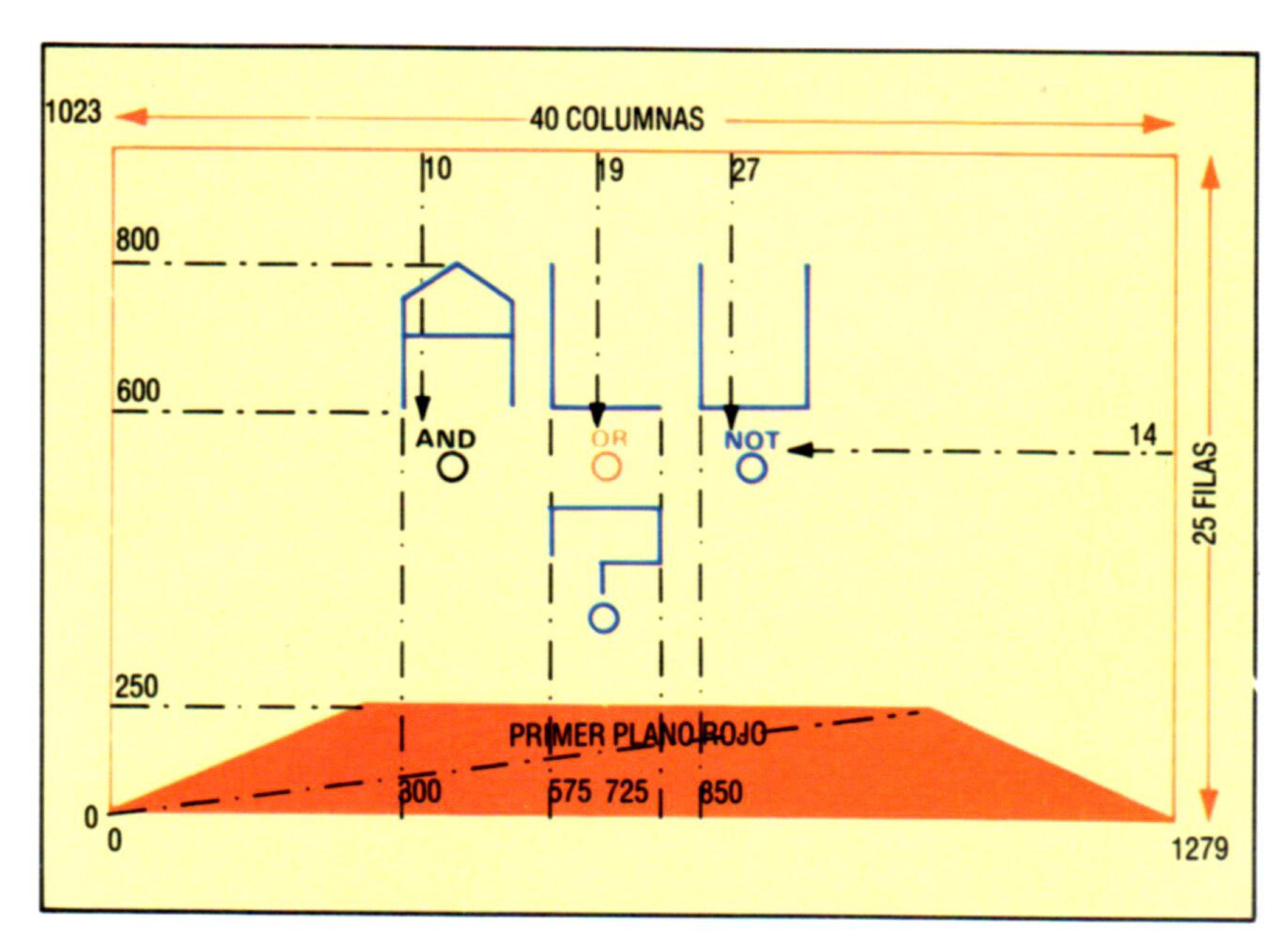
Flujo de información

El baudio, así llamado en honor de J. M. E. Baudot, pionero francés de la transmisión de datos, es una unidad de medida de la velocidad del flujo de información entre dispositivos de comunicaciones. Una velocidad de 1 baudio significa que la información se transmite a 1 bit por segundo. Los dispositivos que se comunican a través de un modem suelen utilizar velocidades de 300 baudios cuando se requiere una transmisión aproximadamente equivalente en ambas direcciones. Cuando la transferencia de datos es básicamente en una sola dirección, como sucede en las comunicaciones tipo videotexto, por ejemplo, se emplea una velocidad más rápida, de 1 200 baudios. Estas velocidades de transmisión son muy similares a la velocidad de transmisión de datos empleada para guardar y cargar programas en cinta

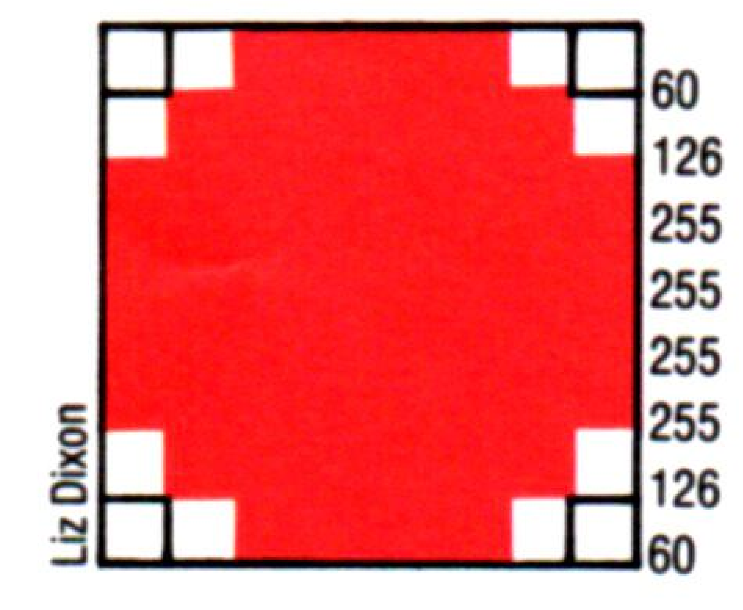
Kevin Jones

Rutinas de pantalla

A lo largo de tres capítulos diseñaremos pantallas de muestra para el BBC Micro, el Commodore 64 y el Spectrum



Mode d'emploi
En un programa para el BBC Micro deben tomarse varias decisiones "comerciales": las modalidades de alta resolución gastan muchísima memoria y soportan pocos colores; las modalidades para texto utilizan menos memoria, permiten mejores gamas de color pero sólo soportan gráficos en resolución media o baja. En este programa, Mode 1 proporciona las resoluciones necesarias, pero al costo de una memoria de pantalla de 20 K



El botón
En la imagen de la ALU para el BBC se requiere una forma de botón para representar las tres opciones, AND, OR y NOT. Es preciso redefinir un carácter ya existente. Utilizando una cuadrícula de 8 por 8 podemos diseñar una forma y representarla mediante 8 números decimales. Entonces se puede redefinir CHR$(240) mediante VDU 23:
VDU 23,240,60,126,255, 255,255,255,126,60

En este capítulo veremos cómo se pueden emplear las facilidades para gráficos del BBC Micro para crear visualizaciones en pantalla para juegos de aventuras. El que hemos venido desarrollando, al que hemos llamado *Digitaya*, es un juego de aventuras basado en texto. Es decir, utiliza palabras para describir los ambientes imaginarios en los cuales se sitúa al jugador. Una aventura basada en texto, por ejemplo, simplemente visualizaría el mensaje "Te hallas en la sala del trono" para evocar un escenario, mientras que una aventura gráfica intentaría dibujar una habitación con un trono.

Las pantallas que diseñaremos aquí visualizarán dos escenarios de *Digitaya* que poseen un interés especial: la entrada a la puerta para palanca de mando y la Unidad Aritmético Lógica (ALU). El número de tales pantallas suele estar limitado por la cantidad de memoria disponible; las instrucciones que se requieren para producir cada visualización ocupan un espacio de memoria que, de otro modo, quedaría libre para aumentar la complejidad de la trama argumental.

Diseño de pantalla de la ALU

Antes de que podamos comenzar a diseñar una pantalla para el BBC Micro, debemos responder a varias preguntas:
1) ¿De cuánta memoria dispongo?
2) ¿Cuántos colores necesito?
3) ¿Cuál es la resolución requerida?
Todas estas preguntas en realidad se pueden fundir en una sola: "¿Qué modalidad utilizaré?" Una resolución mayor y una gama de colores más amplia significan que la zona de memoria para pantalla ocupará una valiosa RAM. En nuestro diseño utilizaremos la modalidad 1, que nos ofrece cuatro colores, una pantalla de 40 por 25 y una resolución media. Hemos de establecer la modalidad a utilizar insertando al comienzo del programa la siguiente línea:

```
1095 MODE 1
```

Una vez decidida la modalidad, podemos hacer un boceto de lo que será nuestra pantalla, trazando a lápiz las coordenadas apropiadas a medida que vamos avanzando. El diseño elegido aquí produce el desplazamiento en la pantalla de las letras A, L y U en mayúscula. En el juego, el jugador debe pulsar uno de tres botones (rotulados AND, OR y NOT) y éstos también deben incluirse en la visualización. Características adicionales incluyen un borde estrecho en todo el margen de la pantalla y un primer plano piramidal. A la izquierda de estas líneas podemos apreciar el aspecto gráfico de nuestro diseño primitivo.

Cada letra se forma con un desplazamiento (MOVE) hasta un punto inicial y utilizando luego PLOT 1 para dibujar la forma de la letra como una serie de líneas relativas al punto inicial. Al diseñar las letras de esta forma, las podemos desplazar por la pantalla simplemente cambiando la instrucción MOVE inicial. Asimismo, podemos borrar las letras volviendo a dibujar sus líneas en la misma posición pero especificando un trazado OR-Excluyente mediante el empleo de GCOL 3.

Los botones se forman redefiniendo un carácter. En este caso, CHR$(240) se redefine mediante el procedimiento *botón* para convertirlo en la forma que vemos abajo a la izquierda. Observe que CHR$(240) es asignado a la variable boton$ para utilizar en la parte principal de la rutina. Los botones y las etiquetas se pueden posicionar sólo con visualizarlos (PRINT) en las coordenadas especificadas en la instrucción TAB.

El primer plano se crea empleando las primitivas de relleno triangular que proporciona la instrucción PLOT 85. Esta instrucción une el punto especificado a los dos puntos trazados previamente y luego llena el triángulo resultante con color. La forma cuadrangular del primer plano se puede dibujar y rellenar mediante dos de estas primitivas de relleno.

El código para la visualización en pantalla constituye una subrutina de la rutina especial diseñada para tratar el escenario ALU del juego. La instrucción A$=GET$, de la línea 7560, espera que se pulse una tecla antes de restaurar el color original del primer plano, limpiando la pantalla y retornando a la rutina ALU principal para seguir con el juego. Para llamar a esta subrutina gráfica, también se debe insertar en el programa principal la siguiente línea:

```
4565 GOSUB 7000: REM S/R IMAGEN ALU
```



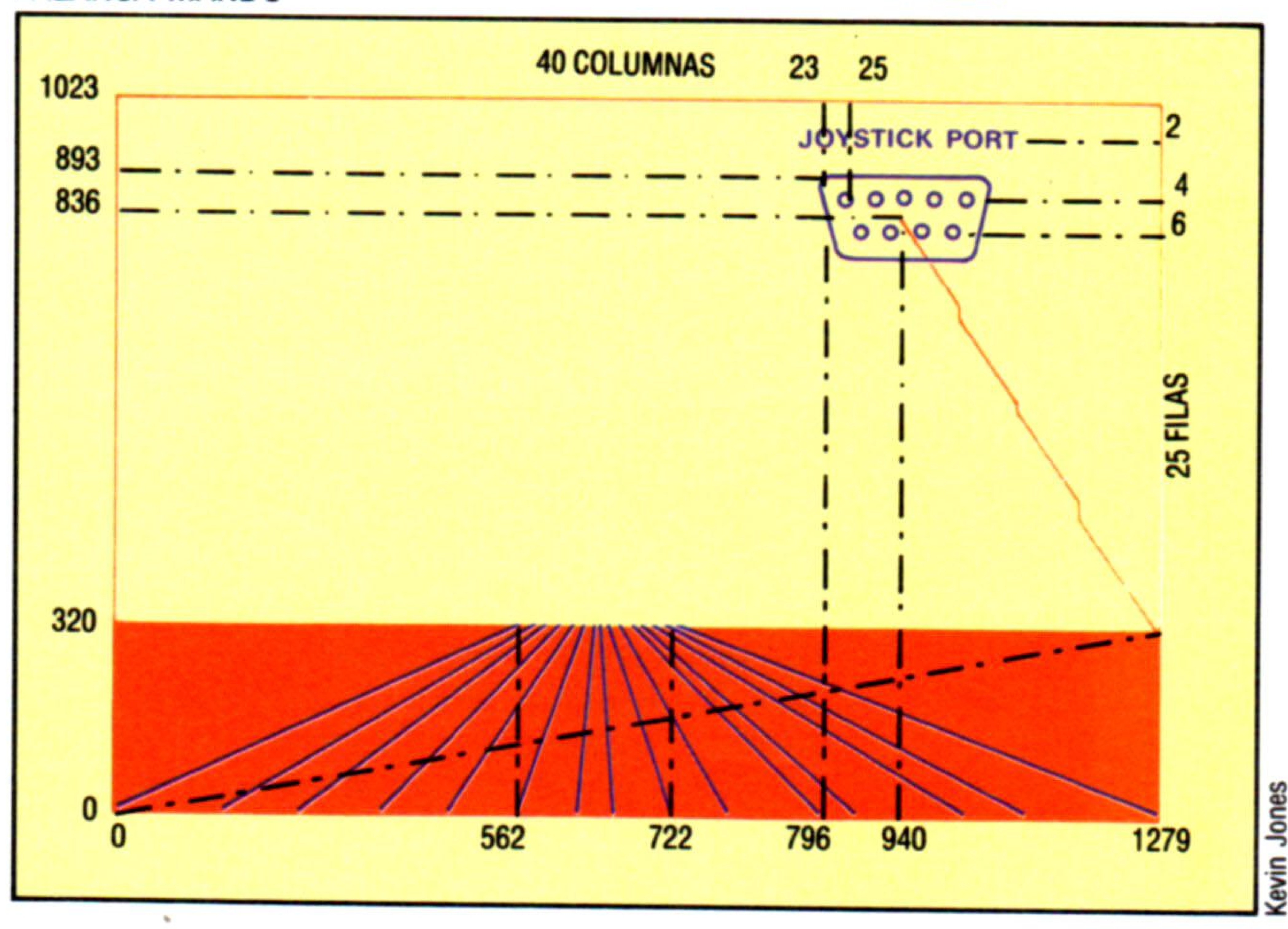
Pantalla de la puerta

En *Digitaya*, si el jugador se extravía en el escenario de la puerta para palanca de mando se encuentra en peligro de ser alcanzado por un rayo láser. El diseño de nuestra visualización en pantalla implica, por consiguiente, dibujar una puerta para palanca de mando de cuyo centro se emitan rayos láser. La puerta para la palanca de mando se dibuja utilizando varios caracteres punto impresos en la esquina superior izquierda de la pantalla, y se dibuja un típico contorno de conector tipo D utilizando gráficos en alta resolución y sentencias PLOT. Observe que después del desplazamiento (MOVE) hasta la posición inicial, todas las sentencias PLOT siguientes que crean la silueta de la puerta son instrucciones PLOT 1, lo que significa que dibujan en relación al último punto trazado. Esto es sumamente conveniente, porque de dibujar las formas empleando una serie de instrucciones relativas, si se decidiera desplazar la posición de la forma completa, sólo debería alterarse la primera sentencia MOVE.

El primer plano está compuesto por un bloque de color rectangular, dibujado empleando nuevamente dos primitivas de relleno triangulares. Para dar la sensación de profundidad, sobre el mismo se dibuja una serie de líneas convergentes utilizando un bucle FOR...NEXT (líneas 8170-8200). El bucle establece el valor de X de 0 a 1280, la anchura de la pantalla en unidades de gráficos. Se trazan hacia la parte inferior de la pantalla una serie de líneas, incrementándose el punto de partida del horizonte para cada punto a medida que se incrementa X. No obstante, en el horizonte el paso de 32, empleado entre líneas consecutivas en la parte inferior de la pantalla, se reduce a un paso de 4 (dividiendo por 8 cada valor de X en la instrucción MOVE que define el punto inicial de cada línea).

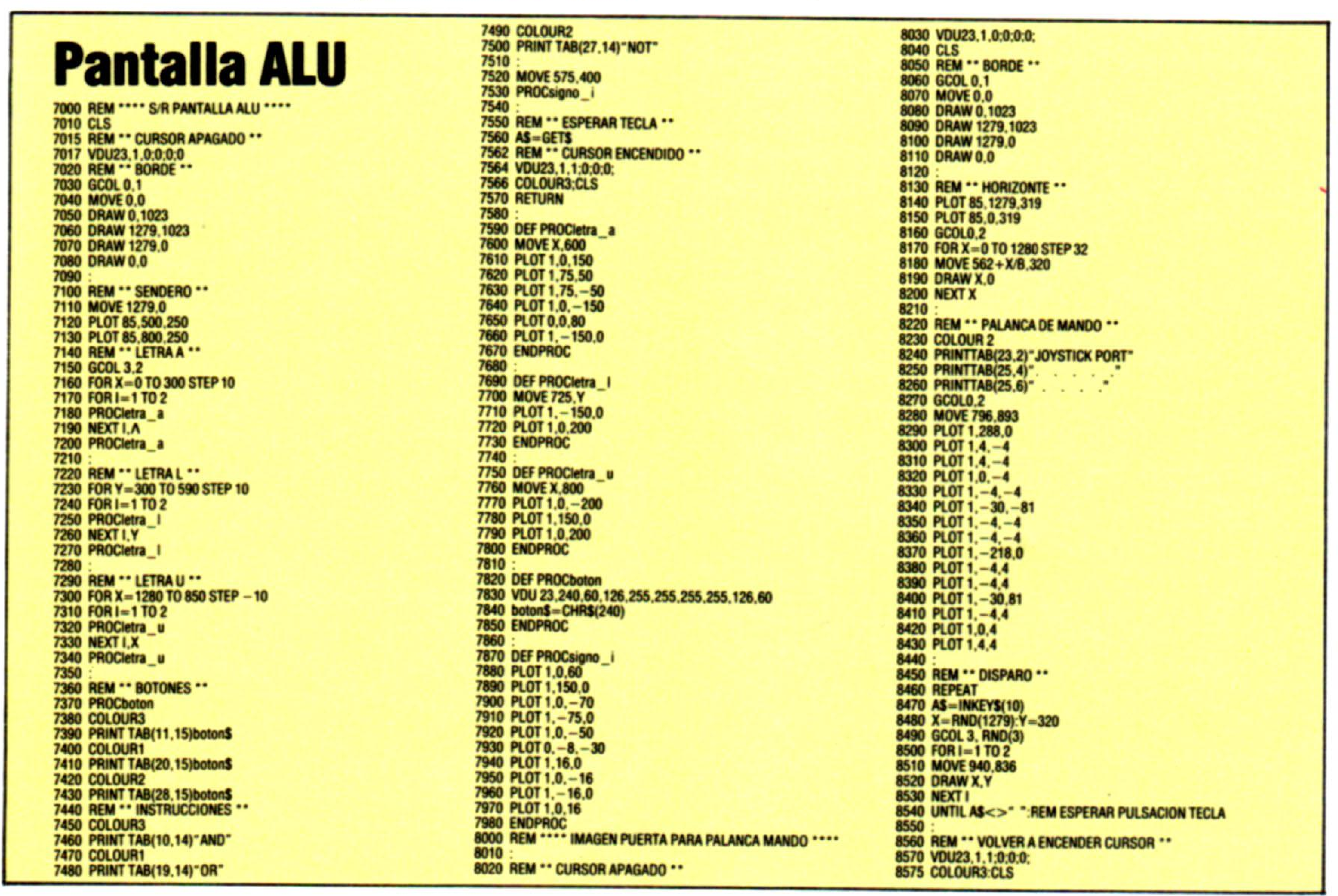
El efecto de los rayos láser se produce dibujando una línea que parte del centro de la puerta para la palanca de mando hasta un punto aleatorio del horizonte y con un color al azar. Posteriormente la línea se borra (sin alterar el fondo) trazando la misma línea en la modalidad de trazado con OR excluyente, establecida mediante GCOL 3. El dibujo y borrado de cada línea están ubicados dentro de un bucle REPEAT...UNTIL, junto con una comparación para comprobar si se ha pulsado alguna tecla en el teclado. El empleo de INKEY$ en lugar de GET$ permite que la ejecución del programa continúe mientras dentro del bucle se está comprobando una posible pulsación de tecla. Este bucle termina cuando se pulsa una tecla; entonces se limpia la pantalla, se restaura el color original del texto y se devuelve el control a la rutina principal de la puerta para palanca de mando. Para llamar a esta subrutina de gráficos se ha de insertar la línea siguiente:

3845 GOSUB 8000: REM IMAGEN PUERTA PARA PALANCA MANDO



Sufriendo un rayo
Tanto en la pantalla de la puerta para la palanca de mando como en la de la ALU se hace un uso exhaustivo de la facilidad para trazado relativo, dado que la misma permite un borrado fácil y el movimiento de formas gráficas completas. Para dibujar (DRAW) y rellenar (FILL) bloques sólidos de color en alta resolución se utiliza otra opción de trazado

Kevin Jones



Pantalla ALU

```
7000 REM **** S/R PANTALLA ALU ****
7010 CLS
7015 REM ** CURSOR APAGADO **
7017 VDU23,1,0;0;0;0
7020 REM ** BORDE **
7030 GCOL 0,1
7040 MOVE 0,0
7050 DRAW 0,1023
7060 DRAW 1279,1023
7070 DRAW 1279,0
7080 DRAW 0,0
7090 :
7100 REM ** SENDERO **
7110 MOVE 1279,0
7120 PLOT 85,500,250
7130 PLOT 85,800,250
7140 REM ** LETRA A **
7150 GCOL 3,2
7160 FOR X=0 TO 300 STEP 10
7170 FOR I=1 TO 2
7180 PROCletra_a
7190 NEXT I,A
7200 PROCletra_a
7210 :
7220 REM ** LETRA L **
7230 FOR Y=300 TO 590 STEP 10
7240 FOR I=1 TO 2
7250 PROCletra_l
7260 NEXT I,Y
7270 PROCletra_l
7280 :
7290 REM ** LETRA U **
7300 FOR X=1280 TO 850 STEP -10
7310 FOR I=1 TO 2
7320 PROCletra_u
7330 NEXT I,X
7340 PROCletra_u
7350 :
7360 REM ** BOTONES **
7370 PROCboton
7380 COLOUR3
7390 PRINT TAB(11,15)boton$
7400 COLOUR1
7410 PRINT TAB(20,15)boton$
7420 COLOUR2
7430 PRINT TAB(28,15)boton$
7440 REM ** INSTRUCCIONES **
7450 COLOUR3
7460 PRINT TAB(10,14)"AND"
7470 COLOUR1
7480 PRINT TAB(19,14)"OR"
7490 COLOUR2
7500 PRINT TAB(27,14)"NOT"
7510 :
7520 MOVE 575,400
7530 PROCsigno_i
7540 :
7550 REM ** ESPERAR TECLA **
7560 A$=GET$
7562 REM ** CURSOR ENCENDIDO **
7564 VDU23,1,1;0;0;0;
7566 COLOUR3;CLS
7570 RETURN
7580 :
7590 DEF PROCletra_a
7600 MOVE X,600
7610 PLOT 1,0,150
7620 PLOT 1,75,50
7630 PLOT 1,75,-50
7640 PLOT 1,0,-150
7650 PLOT 0,0,80
7660 PLOT 1,-150,0
7670 ENDPROC
7680 :
7690 DEF PROCletra_l
7700 MOVE 725,Y
7710 PLOT 1,-150,0
7720 PLOT 1,0,200
7730 ENDPROC
7740 :
7750 DEF PROCletra_u
7760 MOVE X,800
7770 PLOT 1,0,-200
7780 PLOT 1,150,0
7790 PLOT 1,0,200
7800 ENDPROC
7810 :
7820 DEF PROCboton
7830 VDU 23,240,60,126,255,255,255,255,126,60
7840 boton$=CHR$(240)
7850 ENDPROC
7860 :
7870 DEF PROCsigno_i
7880 PLOT 1,0,60
7890 PLOT 1,150,0
7900 PLOT 1,0,-70
7910 PLOT 1,-75,0
7920 PLOT 1,0,-50
7930 PLOT 0,-8,-30
7940 PLOT 1,16,0
7950 PLOT 1,0,-16
7960 PLOT 1,-16,0
7970 PLOT 1,0,16
7980 ENDPROC
8000 REM **** IMAGEN PUERTA PARA PALANCA MANDO ****
8010 :
8020 REM ** CURSOR APAGADO **
8030 VDU23,1,0;0;0;0;
8040 CLS
8050 REM ** BORDE **
8060 GCOL 0,1
8070 MOVE 0,0
8080 DRAW 0,1023
8090 DRAW 1279,1023
8100 DRAW 1279,0
8110 DRAW 0,0
8120 :
8130 REM ** HORIZONTE **
8140 PLOT 85,1279,319
8150 PLOT 85,0,319
8160 GCOL0,2
8170 FOR X=0 TO 1280 STEP 32
8180 MOVE 562+X/8,320
8190 DRAW X,0
8200 NEXT X
8210 :
8220 REM ** PALANCA DE MANDO **
8230 COLOUR 2
8240 PRINTTAB(23,2)"JOYSTICK PORT"
8250 PRINTTAB(25,4)". . . . ."
8260 PRINTTAB(25,6)" . . . ."
8270 GCOL0,2
8280 MOVE 796,893
8290 PLOT 1,288,0
8300 PLOT 1,4,-4
8310 PLOT 1,4,-4
8320 PLOT 1,0,-4
8330 PLOT 1,-4,-4
8340 PLOT 1,-30,-81
8350 PLOT 1,-4,-4
8360 PLOT 1,-4,-4
8370 PLOT 1,-218,0
8380 PLOT 1,-4,4
8390 PLOT 1,-4,4
8400 PLOT 1,-30,81
8410 PLOT 1,-4,4
8420 PLOT 1,0,4
8430 PLOT 1,4,4
8440 :
8450 REM ** DISPARO **
8460 REPEAT
8470 A$=INKEY$(10)
8480 X=RND(1279):Y=320
8490 GCOL 3, RND(3)
8500 FOR I=1 TO 2
8510 MOVE 940,836
8520 DRAW X,Y
8530 NEXT I
8540 UNTIL A$<>" ":REM ESPERAR PULSACION TECLA
8550 :
8560 REM ** VOLVER A ENCENDER CURSOR **
8570 VDU23,1,1;0;0;0;
8575 COLOUR3:CLS
```

Un buen comienzo

Es esencial que el software educativo para niños tenga un buen diseño y claridad en sus objetivos

Los objetivos didácticos de los paquetes que examinaremos abarcan desde el reconocimiento de colores y formas simples, pasando por aptitudes básicas de lectura y numéricas, hasta refinados intentos para ampliar las capacidades artísticas del niño.

Todos los paquetes son bastante directos al explicarle al usuario lo que tiene que hacer. Ésta debe ser una prioridad en todo programa educativo: el niño debe comprender totalmente lo que se espera de él y se le deben dar recompensas claramente definidas una vez que ha conseguido una habilidad.

En segundo lugar, el programa debe ser fácil de utilizar. No tiene sentido que un programa que pretenda proporcionar al niño aptitudes para la lectura empiece con una lista de instrucciones de funcionamiento. Los mejores programas sólo incluyen un mínimo de estas instrucciones.

Un programa educativo debe suscitar el interés del niño. Independientemente de lo importantes o valiosos que sean sus objetivos últimos, no obtendrá ningún resultado si está por encima de las capacidades del niño o si se vuelve repetitivo y aburrido.

Por último, la prueba de fuego de un programa educativo es que enseñe aquello para lo que fue diseñado. Éste parece un punto obvio, pero con frecuencia las firmas de software olvidan los objetivos didácticos de un programa para favorecer su valor como fuente de entretenimiento.

Los paquetes que analizamos aquí los producen las empresas norteamericanas Spinnaker y Fisher-Price. Aunque estos programas en Estados Unidos existen en formato de cartucho, en Europa se comercializan en cassette. Mientras que a un niño pequeño se le puede mostrar claramente cómo debe insertar un cartucho en el ordenador y encenderlo (para que sea capaz, por consiguiente, de cargar por sí mismo su programa), el formato de cassette exige invariablemente que haya cerca un adulto que lo ayude a cargar el programa.

Dance fantasy



De todos los programas que hemos examinado, tal vez el más atractivo sea *Dance fantasy* (Fantasía de la danza), de Fisher-Price, destinado a niños de entre cuatro y ocho años. La visualización en pantalla muestra un escenario en el que hay dos figuras de pie, y se le solicita al usuario que haga la coreografía de un baile para los dos personajes. Antes de comenzar a trabajar, se pide al usuario que especifique el sexo de los bailarines: un varón y una mujer, o dos varones o dos mujeres.

En la parte inferior de la pantalla el programa visualiza una gama de figuras en diversas poses: cada una representa una rutina de danza particular (un salto, una giga, etc.) Moviendo uno de los bailarines sobre una de estas figuras mediante la palanca de mando el niño efectivamente elige esa determinada rutina, y luego vuelve a situar la figura en el escenario, y accionando el pulsador de disparo obtiene su ejecución. De este modo, se puede realizar la coreografía de un baile seleccionando una serie de movimientos y ejecutándolos en diferentes puntos del escenario, proporcionando el programa los movimientos de conexión. Una vez el niño ha completado un baile, puede guardarlo (SAVE) y contemplar después el efecto global.

Aegean voyage



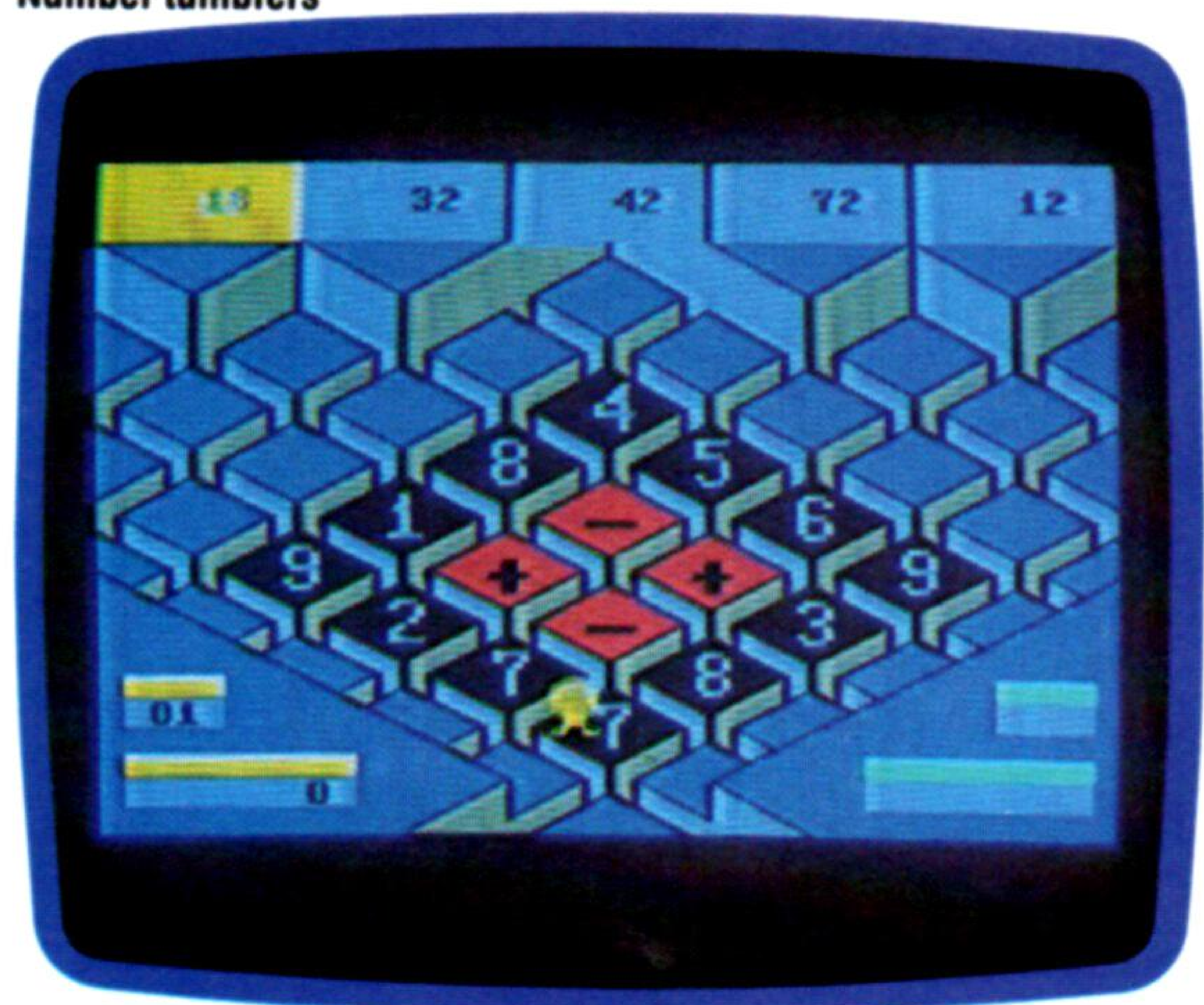
Como habrá observado a partir de esta descripción de *Dance fantasy*, el punto fuerte del programa consiste en que es una imaginativa analogía de lo que es un programa para ordenador: el niño crea su propio baile (programa) utilizando una serie de rutinas básicas (un conjunto de procedimientos). El mismo se guarda (SAVE) luego en cassette y se carga desde el mismo (LOAD), introduciendo de este modo al niño sin ninguna dificultad en el significado de estos dos términos.

Aegean voyage (Viaje por el Egeo), de Spinnaker, dirigido a un grupo de edades ligeramente mayor, emplea personajes y escenarios de la mitología griega como elementos de un sencillo juego

de aventuras. El objetivo consiste en hacer navegar un barco desde Atenas hasta diversas islas del mar Egeo, evitando rocas y tormentas durante la travesía. Al llegar a la seguridad de un puerto, se visualiza el nombre de la isla y en la parte inferior de la pantalla aparece un mensaje críptico. El jugador debe entonces decidir si explora o no la isla: una decisión correcta le será recompensada con un tesoro, como, por ejemplo, el escudo de Aquiles; una decisión incorrecta determinará el hundimiento del barco a manos de una criatura mítica, como, por ejemplo, Gorgona.

A los escolares entendidos en mitología clásica les resultará desconcertante descubrir que en el juego hay datos erróneos: el Minotauro, por ejemplo, es tan probable que aparezca en la isla de Delos como en Creta. El juego no hace ningún intento por explicar el significado de los nombres o de los lugares: es muy improbable que gracias a esta cassette el niño llegue a adquirir siquiera una educación clásica superficial. También es poco probable que el juego logre mantener el interés del niño durante mucho tiempo, porque los gráficos y el formato carecen de interés y son repetitivos.

Number tumblers

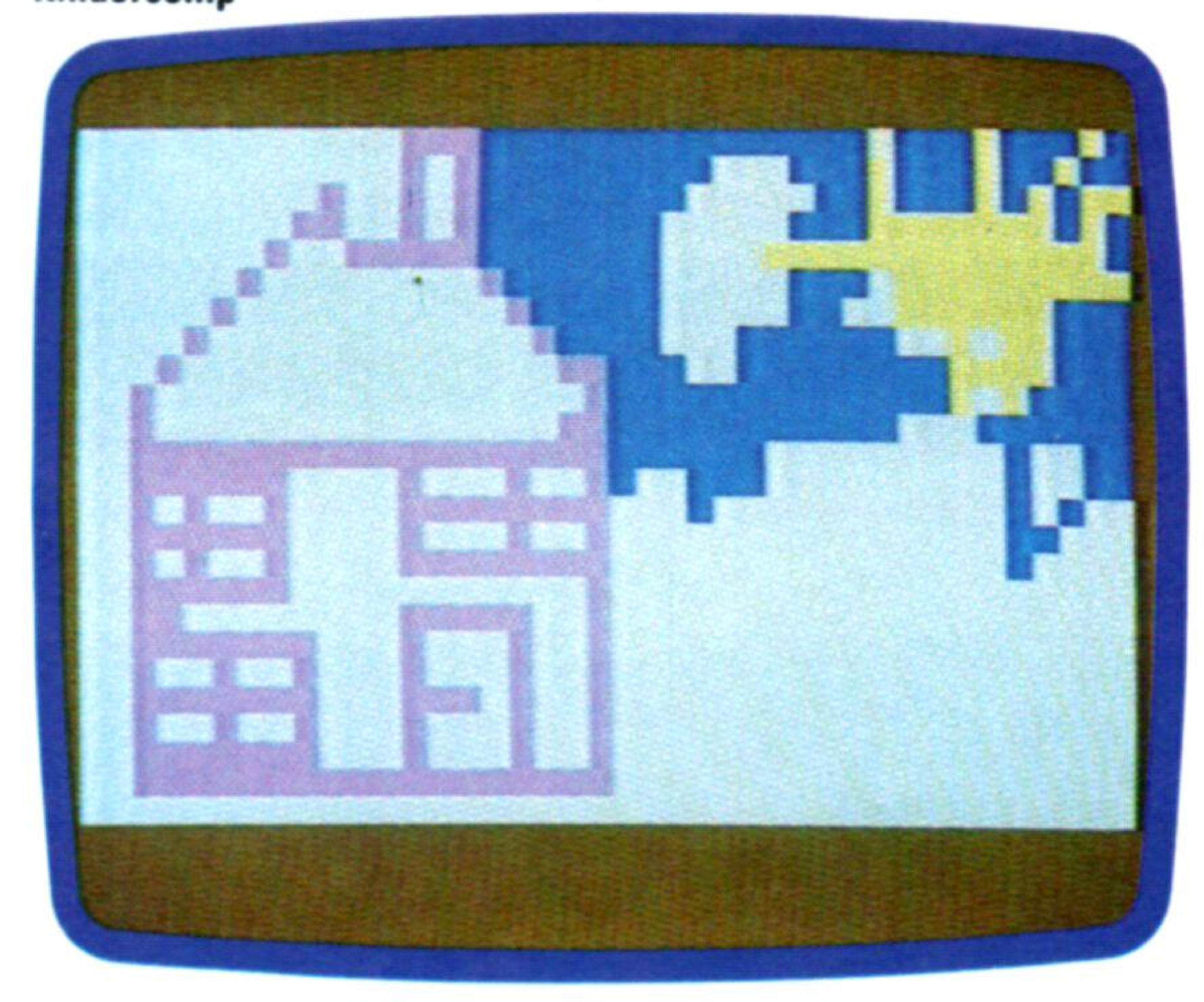


Diseñado para desarrollar aptitudes para la aritmética mental en niños de entre ocho y doce años, *Number tumblers* (Malabaristas de los números), de Fisher-Price, posee la velocidad y las sensaciones de un juego recreativo. En la parte superior de la pantalla se visualiza una serie de números, y el jugador ha de acomodar los símbolos numéricos y aritméticos de las caras de un conjunto de dados para crear una expresión matemática que sea igual a alguno de los números. El juego es rápido, posee gráficos brillantes y bien diseñados y constituirá un auténtico incentivo para que el jugador perfeccione sus aptitudes de cálculo mental.

Kindercomp, de Spinnaker, está destinado a niños de entre tres y ocho años de edad. El objetivo del paquete es introducirlos en el tema de los ordenadores y desarrollar aptitudes artísticas. El paquete se compone de una serie de ejercicios diferentes.

Este paquete lo escribió originalmente el doctor Doug Davis para su hija, presumiblemente como fuente de entrenamiento. Lamentablemente, *Kindercomp* da la sensación de consistir mayormente en hacer trucos con el ordenador, la clase de cosas que diseñan la mayoría de los programadores cuando están aprendiendo BASIC y descubriendo las capacidades de la máquina. Por ejemplo, una de las opciones es "Nombres". Se invita al usuario a entrar un nombre, o una frase corta, de hasta 15 caracteres de longitud, que se visualiza luego a través de toda la pantalla en diversos tamaños y colores. El efecto es muy atractivo y a un niño que no esté familiarizado con los gráficos por ordenador le parecerá asombroso desde el punto de vista visual. Sin embargo, el programa parece tener muy poco valor educativo, puesto que cualquier grupo de letras proporcionará el mismo efecto.

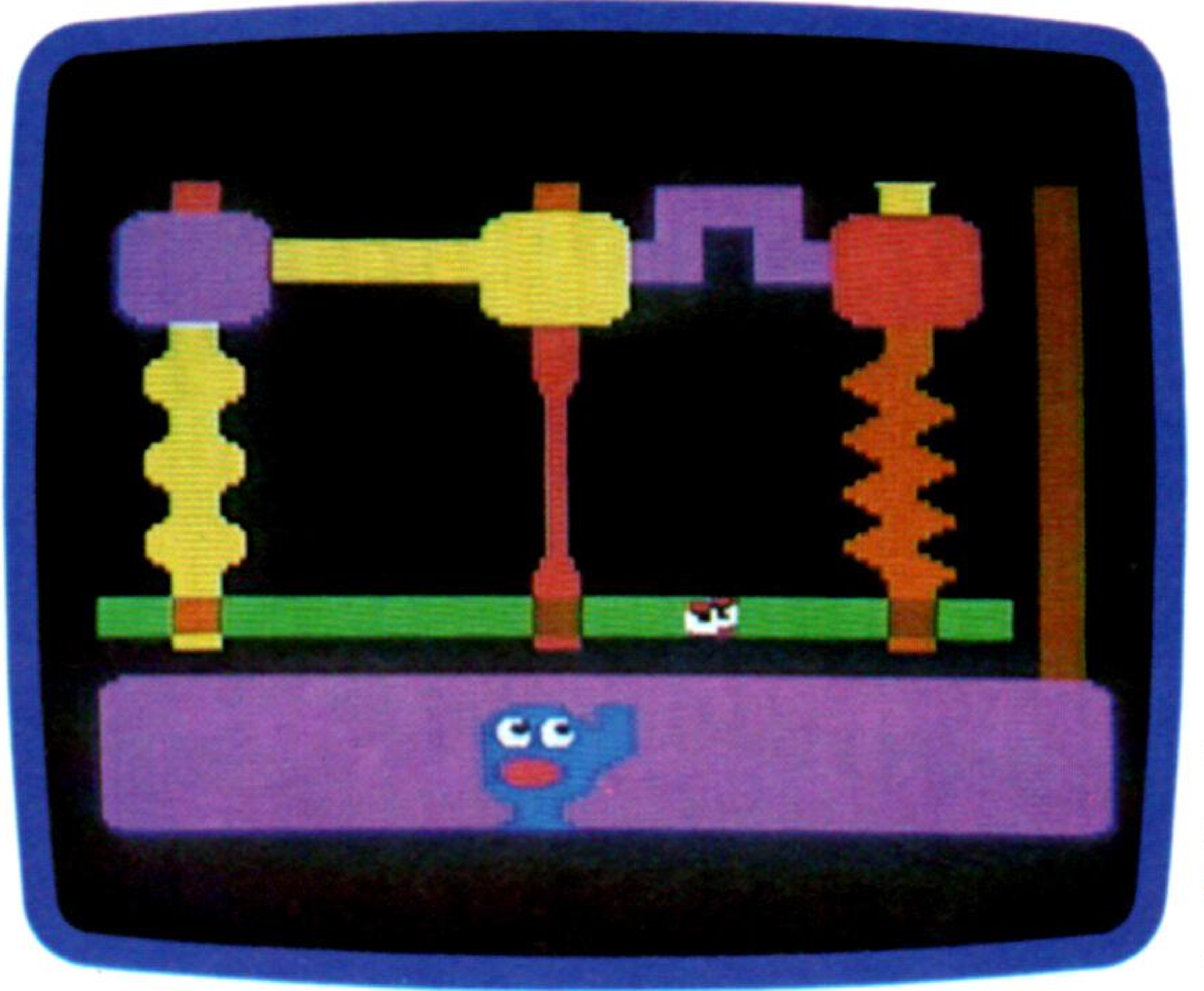
Kindercomp



Es muy poco probable que *Kindercomp* logre mantener al niño ocupado durante mucho tiempo. Los trucos de programación son divertidos pero enseguida se vuelven repetitivos. Es la clase de paquete que con toda seguridad mantendrá al pequeño usuario ocupado durante tres días más o menos y que después jamás se volverá a utilizar.

El último paquete que examinamos está destinado a los niños de muy corta edad. *Alf in the color caves* (Alf en las cuevas de colores), de Spinnaker, caracteriza a un divertido personajillo que se desliza y se escurre por entre numerosos tubos, llenos de color y de diversas formas, de una habitación de la parte inferior de las cuevas. Utilizando una palanca de mando o el teclado, el niño lo va guiando a través de las cuevas y si lo conduce a salvo sin ser detectado por un par de ojos amenazadores que se mueven muy rápidamente, el niño obtiene como recompensa el espectáculo de ver a Alf ejecutando una deliciosa danza. Luego es absorbido por el tubo hacia arriba, nuevamente hasta el nivel del suelo, para iniciar otro descenso.

Alf in the color caves



Ian McKinnell

Misión espacial

He aquí uno de los programas más interesantes escritos para los ordenadores Atari. Utiliza tanto BASIC como lenguaje máquina

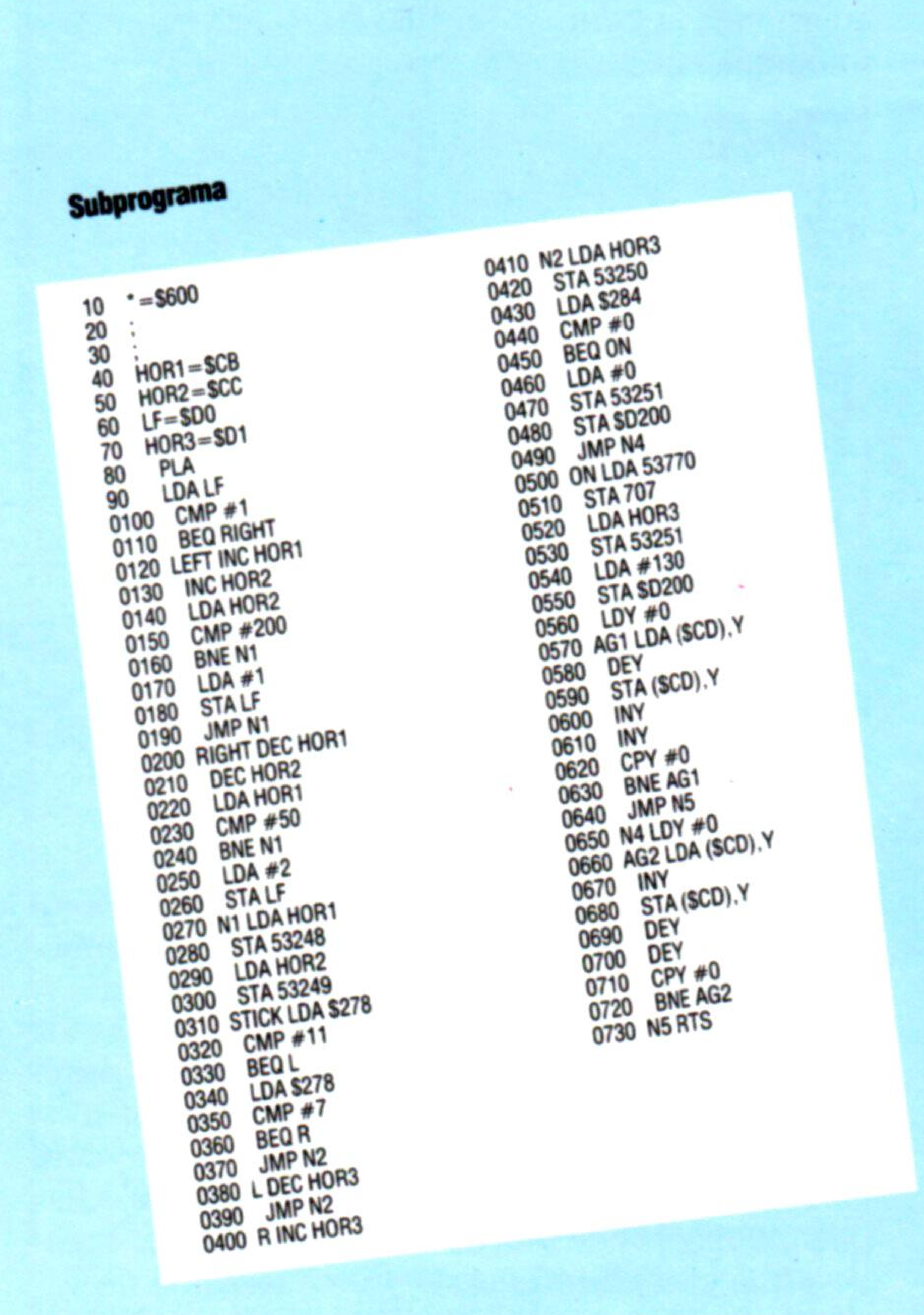


Los movimientos de los diferentes elementos que aparecen en la pantalla están escritos en lenguaje máquina, lo que permite una velocidad superior a la que se obtiene con BASIC. El jugador es el comandante de una nave espacial que se desplaza por una lejana galaxia; dado que la nave ha sido alcanzada por un meteoro, debe, superando cualquier dificultad que se presente, regresar a la base principal para reparar las averías sufridas. ¡Atención!, le quedan sólo 300 l de oxígeno.

Utilice la palanca de mando para conducir su nave. Si tiene éxito en su cometido, la cantidad de aire que quede incrementará su marcador. Quienes se interesen por los subprogramas en lenguaje máquina encontrarán el listado correspondiente a continuación del de BASIC.


```
10 REM ****************************************
20 REM *               MISION                 *
30 REM *              ESPACIAL                *
40 REM ****************************************
50 GOSUB 10000:REM ** INICIALIZACION **
60 GOSUB 2000
65 SOUND 0,0,8,15: POKE 53278,R: AIR=300
70 X=USR(1536)
80 IF PEEK(781+R*256)=24 THEN 1000
90 IF PEEK(53254)<>0 OR PEEK(53262)<>0 THEN
   GOTO 2000
100 AIR=AIR-1:IF AIR<>0 THEN 70
110 POSITION 4,6:? "aire agotado":SC=0
115 POKE 53251,0
120 FOR P=0 TO 4000 STEP 12
130 SOUND 0,P,10,15:NEXT P
140 SOUND 0,0,0,0
150 POSITION 4,6:? "pulse START"
155 POSITION 4,4:? "puntos:";SC
160 POKE 53279,0
170 IF PEEK(53279)<>6 THEN 160
180 POSITION 4,6:? "            ":GOSUB 10100 GOTO
    65
1000 IF PEEK(209)>PEEK(203) AND PEEK(209)<PEEK(204)
     THEN 1020
1010 GOTO 2000
1020 POSITION 4,6:? "bravo"
1025 POKE 53251,0
1030 FOR P=230 TO 10 STEP -10
1040 FOR X=P-5 TO P+5
1050 SOUND 0,X,10,15
1060 NEXT X
1070 SETCOLOR 1,RND(0)*16,10
1080 NEXT P
1090 SOUND 0,0,0,0
1100 POSITION 21,4:? "BONUS EN AIRE:";AIRE
1110 SC=SC+AIR:GOTO 150
2000 FOR P=781 TO 896:IF PEEK(R*256+P)<>24 THEN
     NEXT P
2005 POKE 53251,0
2010 E=53770:POKE P+256*R+RND(0)+8,PEEK(E)
2020 SOUND 0,PEEK(E),8,15:POKE 706,PEEK(E)
2030 S=S+1:IF S<50 THEN 2010
2035 POKE 53250,0:POKE 53251,0
2040 S=0:SC=S:GOTO 140
10000 GRAPHICS 0:DL=PEEK(560)+256*PEEK(561)
      +4:POKE 82,0
10010 POKE DL+5,7:POKE DL+6,6:POKE DL+7,6:POKE
      DL+8,6:POKE DL+9,6
10020 SETCOLOR 4,9,4:SETCOLOR 1,3,4:POKE DL+10,6
10030 SETCOLOR 0,3,8:SETCOLOR 1,13,10
10040 POSITION 0,4:POKE 752,1
10050 ? " MISION ESPACIAL ";
10055 ? " *************** ";
10060 ? "estas perdiendo aire ";
10070 ? "y DEBES reunirte ";
10080 ? "con la nave madre ";
10090 ? " ...BUENA SUERTE!!! ";
10100 R=PEEK(106)-8:RESTORE
10110 FOR P=R*256+512 TO R*256+1024
10120 POKE P,0:NEXT P
10130 FOR P=10 TO 17
10140 READ A,B
10150 POKE R*256+512+P,A
10160 POKE R*256+640+P,B
10170 NEXT P
10180 DATA 7,224,62,124,111,246
10190 DATA 255,255,214,107
10200 DATA 124,62,60,60,4,32
10210 FOR P=0 TO 7
10220 READ A
10230 POKE R*256+768+80+P,A
10240 NEXT P
10250 DATA 24,60,36,60,36,126,90,255
10260 FOR P=0 TO 6
10270 READ A
10280 POKE R*256+896+88+P,A
10290 NEXT P
10300 ? CHR$(125):COLOR 160:PLOT 0,20:DRAWTO 39,20:?
      CHR$(160):POSITION 0,0
10310 POKE 704,8*16
10320 POKE 705,8*16
10330 POKE 706,15*16+10
10340 POKE 707,3*16+4
10350 POKE 53277,3
10360 POKE 559,46
10370 POKE 53248,110
10380 POKE 53249,126
10390 POKE 53250,108
10400 POKE 53256,1
10410 POKE 53257,1
10420 POKE 53258,1
10430 POKE 53259,1
10440 POKE 623,1
10450 POKE 54279,R
10460 DATA 24,60,126,255,255,0,0
10470 POKE 203,110:POKE 204,126
10480 POKE 208,INT(RND(1)*2)+1:POKE 209,108
10485 K=R*256+768:POKE 206,INT(K/256):POKE
      205,K-(256*PEEK(206))
10490 RETURN
20000 FOR A=1536 TO 1676:READ B:POKE A,B:NEXT A
20001 DATA 104,185,208,201,1,240,17,230,
      203,230,204,165,204,201,200,208,21,169,1,
      133,208,76,38,6,198
20002 DATA 203,198,204,165,203,201,50,20,8,
      4,169,2,133,208,165,203,141,0,208,165,204,
      141,1,208,173,120
20003 DATA 2,201,11,240,10,173,120,2,201,
      7,240,8,76,72,6,198,209,76,72,6,230,209,165,
      209,141
20004 DATA 2,208,173,132,2,201,0,240,11,
      169,0,141,3,208,141,0,210,76,127,6,173,10,
      210,141,195
20005 DATA 2,165,209,141,3,208,169,130,
      141,0,210,160,0,177,205,136,145,205,200,200,
      192,0,208,245,76
20006 DATA 140,6,160,0,177,205,200,145,205,
      136,136,192,0,208,245,96
20007 RETURN
```

Subprograma



```
10    *=$600
20    ;
30    ;
40    HOR1=$CB
50    HOR2=$CC
60    LF=$D0
70    HOR3=$D1
80      PLA
90      LDA LF
0100    CMP #1
0110    BEQ RIGHT
0120 LEFT INC HOR1
0130    INC HOR2
0140    LDA HOR2
0150    CMP #200
0160    BNE N1
0170    LDA #1
0180    STA LF
0190    JMP N1
0200 RIGHT DEC HOR1
0210    DEC HOR2
0220    LDA HOR1
0230    CMP #50
0240    BNE N1
0250    LDA #2
0260    STA LF
0270 N1 LDA HOR1
0280    STA 53248
0290    LDA HOR2
0300    STA 53249
0310 STICK LDA $278
0320    CMP #11
0330    BEQ L
0340    LDA $278
0350    CMP #7
0360    BEQ R
0370    JMP N2
0380 L DEC HOR3
0390    JMP N2
0400 R INC HOR3
0410 N2 LDA HOR3
0420    STA 53250
0430    LDA $284
0440    CMP #0
0450    BEQ ON
0460    LDA #0
0470    STA 53251
0480    STA $D200
0490    JMP N4
0500 ON LDA 53770
0510    STA 707
0520    LDA HOR3
0530    STA 53251
0540    LDA #130
0550    STA $D200
0560    LDY #0
0570 AG1 LDA ($CD),Y
0580    DEY
0590    STA ($CD),Y
0600    INY
0610    INY
0620    CPY #0
0630    BNE AG1
0640    JMP N5
0650 N4 LDY #0
0660 AG2 LDA ($CD),Y
0670    INY
0680    STA ($CD),Y
0690    DEY
0700    DEY
0710    CPY #0
0720    BNE AG2
0730 N5 RTS
```



Mirada certera

Las más modernas cámaras incorporan microprocesadores, que se encargan de los detalles técnicos necesarios para obtener una buena fotografía

Cuando hace una fotografía, la primera tarea del fotógrafo es decidir la *exposición* correcta. Ello implica determinar cuánta luz de una escena en especial llegará hasta la película de la cámara: si es demasiada, la fotografía quedará descolorida; si es muy escasa, ésta será tan oscura que no se distinguirá nada. Para conseguir la exposición correcta es necesario hallar un equilibrio entre la *apertura* (el tamaño del agujero de la lente, que determina cuánta luz penetra) y la *velocidad de obturación*, que determina la longitud de la exposición.

Por consiguiente, la cantidad de luz de una escena en particular se debe primero medir y, teniendo en cuenta la sensibilidad de la película, determinar en consecuencia la apertura y la velocidad de obturación. Con el correr de los años se han desarrollado medidores que permiten al fotógrafo efectuar una medición fiable del brillo de una escena. Más recientemente se han incorporado a las cámaras medidores de exposición, si bien el fotógrafo todavía debe seleccionar una velocidad de obturación y la apertura de acuerdo a la lectura del medidor.

Los avances que experimentó la electrónica en los años setenta ha hecho posible traducir la lectura del medidor de luz directamente en puntos para la apertura o la obturación. Ello se realiza sin que el fotógrafo intervenga en absoluto, de modo que se pueden obtener resultados de buena calidad simplemente dirigiendo la cámara hacia un determinado punto y disparando. Esta facilidad es particularmente útil tanto para el principiante, que desea tomar fotografías sin entender cómo funciona la máquina, como para los fotógrafos profesionales, que muchas veces necesitan hacer fotografías en condiciones inadecuadas.

La Canon A1 ofrece seis modalidades diferentes para hacer fotografías. Éstas son:

1) *Prioridad a la obturación:* El usuario selecciona una velocidad de obturación y la cámara establece la apertura correspondiente.
2) *Prioridad a la apertura:* El usuario selecciona una apertura y la cámara establece la velocidad de obturación.
3) *Programa:* La cámara establece tanto la velocidad de obturación como la apertura mediante un programa que las combina óptimamente.
4) *Flash automático:* Si está equipada con ciertos flashes, la cámara establece de forma automática la velocidad de obturación adecuada para flash (1/60 segundo) y establece la apertura correspondiente para éste. Un medidor de luz situado en el flash reduce el poder de éste cuando ya ha rebotado suficiente luz desde el tema.



Ian McKinnell

Cámaras eficientes
Desde los años setenta los microprocesadores han venido controlando las funciones interrelacionadas de la medición de luz y el punto de apertura. Las cámaras controladas por microprocesador de hoy en día son capaces de mucho más: se ocupan de la obturación, la exposición y las funciones del flash para permitir que un usuario novato realice tomas de gran calidad, aun en condiciones de luz desfavorables. Dos de tales cámaras son la Nikon FA y la Pentax Super A, capaces de efectuar programas alternativos de control que abarcan una gama de situaciones diferentes

5) *Prioridad de apertura diafragmada:* Es para lentes de tipo antiguo y ciertos accesorios que funcionan con la apertura de la lente siempre "diafragmada" al valor utilizado para tomar la fotografía. Las lentes modernas permanecen en el punto máximo de apertura, excepto en el instante en que se toma la fotografía, para mantener la imagen del visor lo más brillante posible.
6) *Manual:* El fotógrafo establece tanto la velocidad de obturación como de apertura. Es útil para obtener un control total cuando se desea un efecto especial o la iluminación es extraña.

Una visualización electrónica dentro del visor de la Canon A1 indica al fotógrafo en qué modalidad está la cámara y qué valores ha establecido para la obturación y la apertura. La visualización emplea LED para que sea visible incluso en la oscuridad.

A pesar de la utilidad general de la modalidad programa de la Canon A1, ésta funciona de acuerdo a una fórmula bastante simple. Esto significa que, en unos pocos casos, no selecciona la mejor combinación posible. Por ejemplo, si se estuvieran sacando fotografías a última hora de la tarde, la cámara podría seleccionar una velocidad de obturación de 1/30 segundo y una apertura de f/2.8. Las fotografías tomadas a una velocidad de obturación inferior a 1/60 corren el riesgo de arruinarse por los imperceptibles movimientos de la mano del fotógrafo (lo que se conoce como "temblor de la cámara"). Cuando la velocidad de obturación desciende de 1/60 segundo, la cámara hace centellear un aviso previniendo del peligro del temblor de la cámara, pero el programa no selecciona una apertura mayor para permitir la mayor velocidad de obturación.

La competencia de Canon

Algunas empresas rivales han lanzado cámaras de modalidades múltiples con microprocesadores incorporados. La Pentax Super A utiliza un programa ligeramente más sofisticado que la Canon A1, que le proporciona una mejor combinación de velocidades de obturación y aperturas tanto con luz intensa como con luz tenue. En la situación de "últimas horas de la tarde" que describíamos antes, la Pentax Super A seleccionaría una velocidad de poco menos de 1/60 segundo, de modo que habría menos posibilidades de que se produjera un temblor de cámara. Y la Nikon FA, fabricada por el rival por excelencia de Canon, detecta automáticamente cuándo hay instalado un teleobjetivo (longitud focal de 135 mm o más) y accede a un programa alternativo. Éste está optimizado para evitar el temblor de cámara con una lente mayor mediante el empleo de velocidades de obturación más rápidas y mayores aperturas.

Haciéndose eco de la tendencia de Nikon, Canon ha utilizado tres programas alternativos en su última cámara, la Canon T70. Uno es para lentes normales, otro para teleobjetivos y el tercero para lentes granangulares. No obstante, la cámara no reconoce automáticamente cuál es la lente instalada, por lo que el usuario debe seleccionar el programa apropiado. Esto no representa necesariamente un inconveniente, puesto que permite un pequeño control creativo adicional. Por ejemplo, si usted está disparando hacia un punto que se mueve rápidamente con una lente granangular, puede seleccionar la modalidad de teleobjetivo para obtener grandes velocidades de obturación y asegurar que la acción se congele.

Otro problema de las cámaras automáticas es que dan una exposición promedio para toda la imagen, y los objetos con una extremada gama de brillo en la imagen pueden engañar fácilmente al medidor. Por ejemplo, si se fotografía una motocicleta contra un fondo de atardecer, la cámara tenderá a dar la exposición correcta para el sol y hará que la moto quede demasiado oscura. Por otra parte, si la motocicleta se fotografiara contra un fondo oscuro, la cámara trataría la escena como si fuera mucho más oscuro de lo que en realidad es y probablemente la fotografía quedaría sobreexpuesta.

La Nikon FA utiliza una novedosa forma de abordar el problema. En vez de tomar una sola medida del brillo de una escena, mide cinco partes diferentes de la misma. La FA emplea luego un microprocesador para comparar las cinco lecturas con varias "escenas estándares" programadas en la cá-

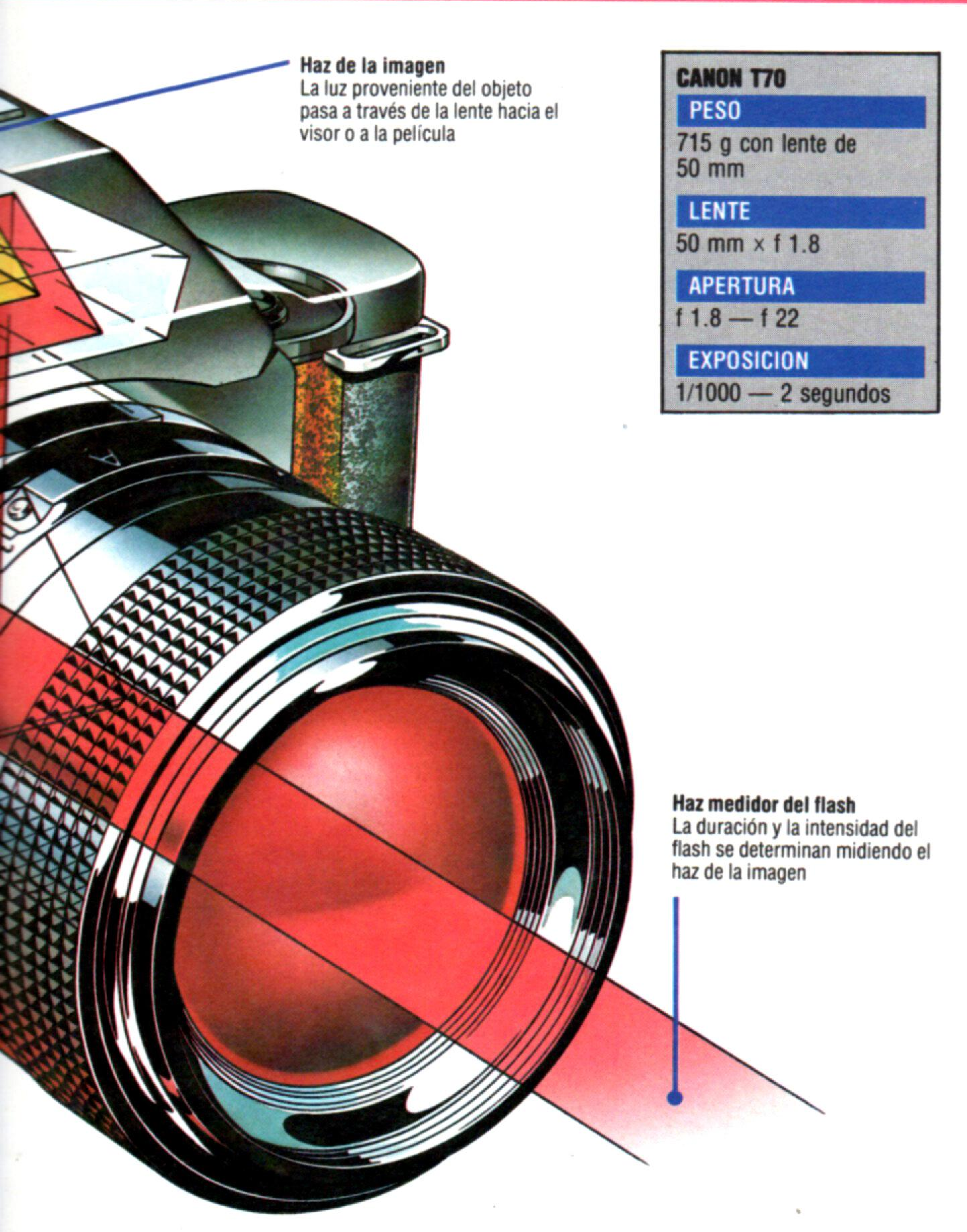
Imagen del visor
Al utilizar el haz de imagen, el visor da una imagen "a través de la lente" (reflex)

Pantalla LCD en modalidad programa
La Canon T70 posee tres programas para exposición-apertura seleccionados por el usuario (para lentes normales, teleobjetivos y granangulares), más opciones manual y semiautomática

Espejo con resortes
Dirige el haz de la imagen hacia el prisma del visor hasta que se acciona el pulsador de obturación

Microfotos

Una CPU de ocho bits construida especialmente controla la operación global de la Canon T70, con la ayuda de chips de medición y de ISO. El oscilador temporizador de cristal genera los impulsos síncronos del reloj y controla la longitud de la exposición. Unos contactos controlados por el medidor IC establecen la apertura. Se puede incorporar un módulo opcional de instrucciones que ofrece exposición automática a intervalos (entre un segundo y un día) y permite escribir directamente en el negativo los datos de temporización



CANON T70
PESO
715 g con lente de 50 mm
LENTE
50 mm × f 1.8
APERTURA
f 1.8 — f 22
EXPOSICION
1/1000 — 2 segundos

mara. Cada una de estas escenas se produce analizando centenares de fotografías.

Pero de todas las cámaras disponibles en la actualidad, la que hace el uso más cabal de la electrónica es la Canon T70. La T70 no posee controles mecánicos; todos sus ajustes se efectúan accionando pulsadores. Se puede seleccionar una de sus ocho modalidades accionando un pulsador situado en la parte superior izquierda de la cámara. Éste hace que aparezca información en una gran LCD situada en la parte superior derecha de la cámara. La información importante, como la velocidad de obturación, la apertura y la modalidad, también aparece en el visor, de manera que el fotógrafo al encuadrar una fotografía no tiene que alejar la cámara de su ojo.

La Canon T70 en acción

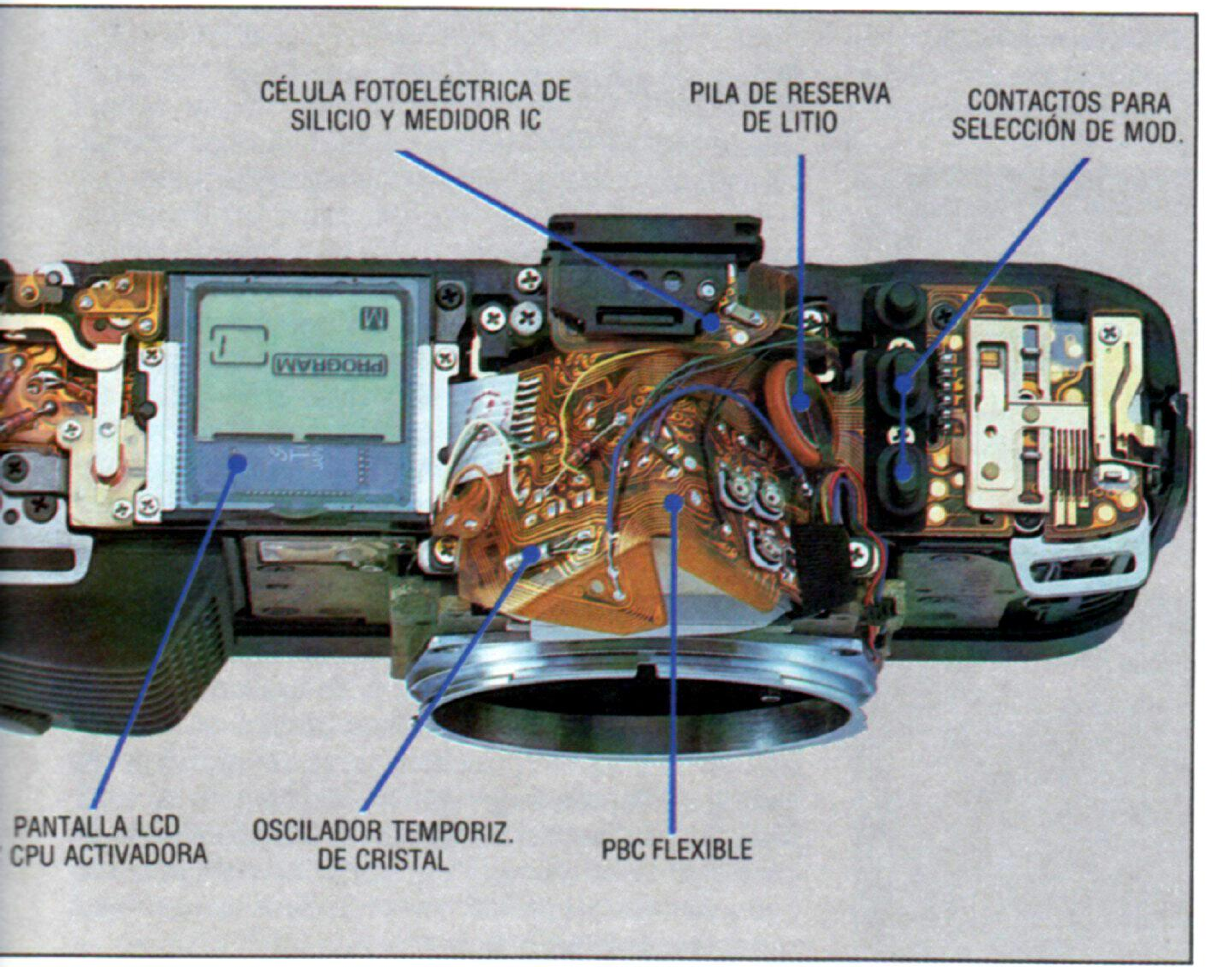
Cuando se selecciona una velocidad de obturación, los ajustes se efectúan en relación al valor actual visualizado en la LCD, mediante dos pulsadores que aumentan o disminuyen la velocidad. La sensibilidad de la película (conocida como su número de ASA o ISO) se establece de forma muy similar. El contador de disparos, que muestra cuántas fotografías se han tomado, también aparece en la visualización en cristal líquido.

La máquina posee un motor incorporado para avanzar la película y rebobinarla cuando se termina el carrete. La T70 funciona con dos pilas normales, y en la LCD hay tres barras que indican su estado. Si aparecen las tres barras, las pilas están nuevas; dos barras señalan que están parcialmente agotadas, y una barra avisa que es necesario sustituirlas. Si se utiliza el disparador automático, una visualización en la LCD cuenta hacia atrás los segundos hasta liberar el obturador.

Los microprocesadores que se utilizan en las cámaras son mucho menos potentes que los que se emplean en los ordenadores. La T70 posee su propio microprocesador de ocho bits de fabricación especial, que es de tipo CMOS para mantener al mínimo su consumo de energía. Opera a una velocidad de reloj de sólo 32 KHz; los microordenadores trabajan aproximadamente 100 veces más rápido. Cuando no se la está utilizando, la cámara pasa a unos magros 8 KHz para economizar potencia.

El microprocesador está alojado en un paquete plano de 60 patillas y posee una gran cantidad de ROM pero sólo 16 K de RAM. Con el microprocesador trabajan otros cuatro chips, siendo el más importante el de entrada/salida. Éste controla la operación mecánica de la cámara gracias a unos imanes y un motor. Asimismo, convierte la señal eléctrica analógica del chip del medidor de luz en una señal digital que pueda ser comprendida por el microprocesador.

El poder de la microelectrónica hace posible disfrutar enormemente captando fotografías de gran calidad sin tener que comprender las complejidades de la fotografía. Aun así, siempre habrá casos en los cuales la cámara dará una exposición incorrecta o enfocará el objeto equivocado. La persona que realmente entienda el funcionamiento de una cámara siempre estará en una situación ventajosa respecto a un principiante que posea un equipo fotográfico sofisticado, pero año tras año esta ventaja se está reduciendo cada vez más.



Fuente de energía

En esta ocasión analizaremos los servomotores y sugeriremos algunos usos a los que se pueden destinar

Existen tres tipos de motores eléctricos: de corriente directa, paso a paso y servo. Un motor de *corriente directa* (CD) se puede controlar fácilmente por ordenador, pero tiende a ser impreciso si encuentra cualquier resistencia. En tal situación se reduce la velocidad del motor y el ordenador no puede mantenerse informado sobre su posición.

Un motor *paso a paso* no tiene este problema, porque se mueve en intervalos de ángulo fijo (por ejemplo, 7,5°) cada vez que se le da un único impulso de corriente. Contando los impulsos, y suponiendo que el motor no esté nunca sobrecargado, el ordenador puede calcular la posición del motor. Los motores paso a paso se utilizan mucho en los sistemas controlados por ordenador, como brazos-robot, tornos, distribuidores, etc. Sin embargo, los impulsos de control también le distribuyen energía al sistema y, por lo tanto, los motores requieren activadores construidos especialmente.

En las tiendas de maquetas y modelos a escala se pueden adquirir pequeños *servomotores* digitales, ya que éstos se utilizan comúnmente para los aviones, barcos, coches, etc., controlados por radio. El tamaño de estos motores varía desde aproximadamente la mitad de una caja de fósforos hasta casi diez veces ese tamaño. Algunos servomotores son sumamente fuertes y capaces de proporcionar más "momento de torsión" (término que se utiliza para describir cualquier fuerza que provoque una rotación) del que podrían producir la mayoría de las personas empleando un destornillador grande. Incluso los motores más baratos de este tipo son adecuados para hacer brazos-robot pequeños, etc.

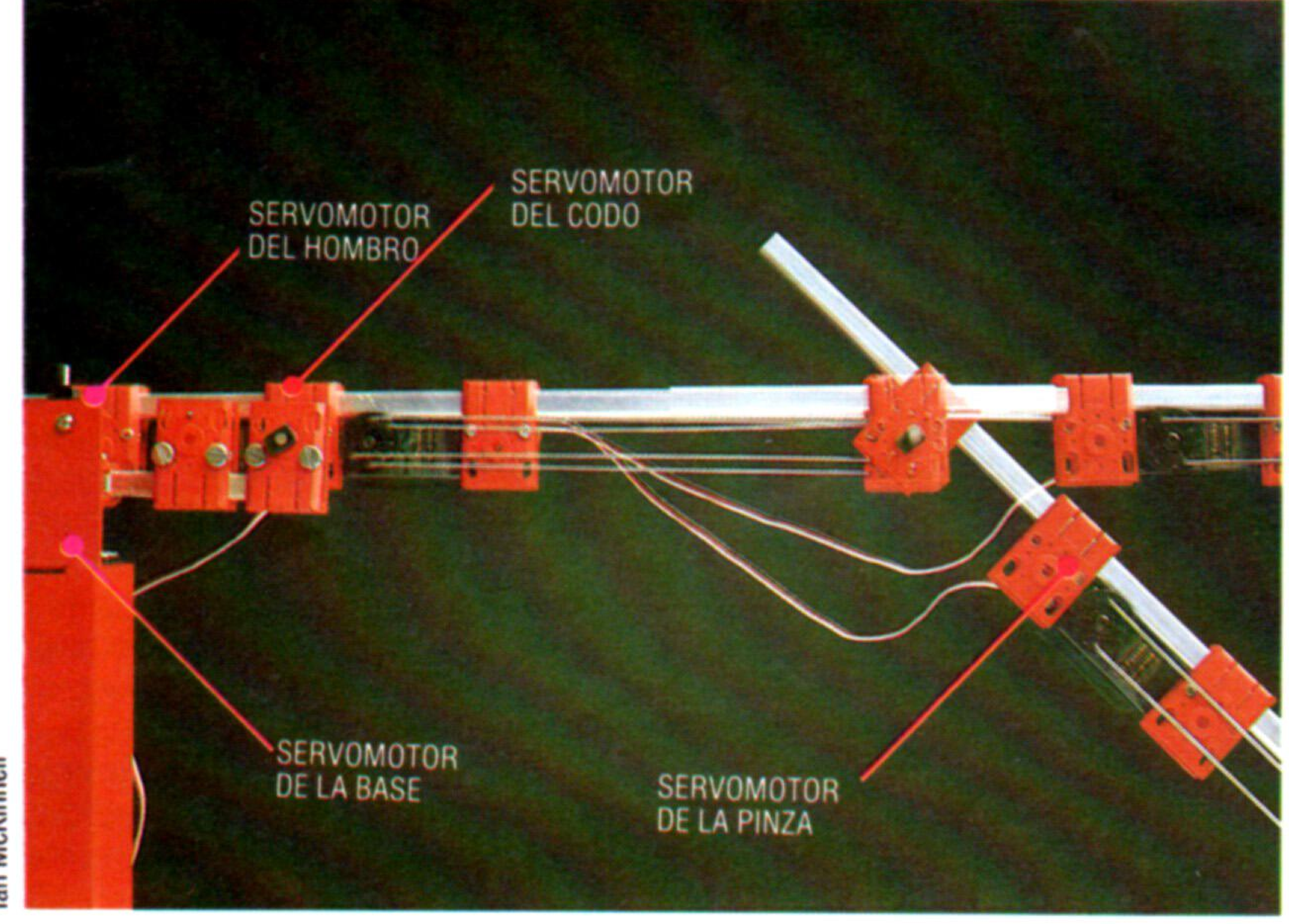
Un servomotor digital pequeño, como un Futuba FP-S126 o un Axoms AS-1, contiene un potenciómetro de realimentación y un diminuto motor CD enlazado (mediante una serie de engranajes) a un "asta". Esta última es una prominencia del motor en la cual se pueden fijar palancas, dientes de engranaje, etc. El motor también es ideal para montar un sistema de bucle de realimentación con un ordenador, ya que la caja del motor también contiene todo el sistema de circuitos necesarios para hacerlo, así como un chip controlador del circuito integrado.

Un servomotor digital típico de modelista se alimenta desde una fuente de 5 V y su ángulo (posición) se establece a través de un cable de control separado. Un impulso de un milisegundo (1/1000) desplazará el asta en una dirección, mientras que un impulso de dos milisegundos la moverá en la otra dirección, describiendo el mismo ángulo. Las variaciones en el ángulo de movimiento son proporcionales a la duración del impulso.

Sin embargo, el motor permanecerá activo sólo durante unos 20 milisegundos después del impulso, momento en que se "relaja" y retorna a su posición original. Por consiguiente, para mantener la palanca en un ángulo determinado, el impulso de control debe repetirse a una frecuencia de unos 50 Hz.

Los servomotores se suelen emplear para mover palancas, etc., pero también pueden ser utilizados para el movimiento lineal. Si el potenciómetro se desacopla del tren de engranajes y se centra, el motor girará continuamente. De hecho, la velocidad a la cual gira está determinada por la duración del impulso.

Bestia de carga
El brazo-robot Beasty está alimentado por tres servomotores (giros de base, hombro y codo) con un cuarto servomotor opcional para activar el efector final. El servomotor es ideal para el brazo-robot, precisamente porque se lo puede mover y después fijar en esa posición



Ian McKinnell

Conexión al ordenador

Cuando los servomotores se conectan directamente a la puerta para el usuario de un ordenador, los errores en el cableado pueden dañar los delicados mecanismos internos de la máquina. Por lo tanto, se ha de utilizar un sistema de tamponamiento, y las cajas buffer y de salida que construimos anteriormente en este apartado son ideales para este fin. Usted debe conectar el cable de la fuente de alimentación eléctrica del motor al conector positivo (rojo) de una de las líneas de la caja de salida de bajo voltaje. El cable a tierra común se debe conectar al conector negativo (negro).

Si el cable de control del motor se ha conectado a la línea de datos 0 de la puerta para el usuario del ordenador, entonces se puede controlar el motor propiamente dicho enviando a la línea de datos 0 de la puerta para el usuario las secuencias de impulsos apropiadas. Un impulso se envía elevando la línea 0 a 5 V, mediante el almacenamiento de un 1 binario en la línea 0. Se utiliza entonces un bucle de cuenta hacia atrás para esperar el tiempo deseado antes de volver a reducir la salida mediante el almacenamiento de un 0 binario en la línea 0.

Tanto el BBC Micro como el Commodore 64 utilizan chips adaptadores para interface versátiles para conformar la puerta para el usuario. Dado que una puerta para el usuario se puede emplear tanto para entrada como para salida, la que estamos utilizando se debe establecer primero en la modalidad requerida (en este caso, para salida). En ambos micros esto se puede realizar manipulando (POKE) directamente el registro de control de dirección de datos usando BASIC.

Ahora debemos considerar cómo enviar impulsos por una línea de datos de la puerta para el usuario. Si se colocara (POKE) en ésta un valor de 88 hexadecimal (equivalente al 136 decimal, o a un patrón de bits binario de 10001000), los voltajes de las patillas de datos serían: 5 V, 0 V, 0 V, 0 V, 5 V, 0 V, 0 V, 0 V respectivamente. Este patrón permanecería constante hasta que fuera modificado deliberadamente. Por consiguiente, se puede generar un impulso simple en la línea de datos 0 colocando (POKE) hexa 00, hexa 88, hexa 00.

Para que el impulso sea lo bastante rápido, se debe utilizar código máquina. El algoritmo para enviar impulsos a un único servomotor es:

1) Especificar el ángulo de la palanca almacenándolo en un solo byte (llamado ANGULO) con un valor entre 0 y 255, utilizando un programa en BASIC.
2) Establecer alta (5 V) la línea de datos 0, comenzando de este modo el impulso.
3) Esperar un milisegundo mediante bucle y decremento del contador.
4) Esperar durante otro período de entre 0 y 1 milisegundo, nuevamente mediante bucle, pero esta vez el valor de partida del contador (y, por tanto, la cantidad de iteraciones) es ANGULO.
5) Poner baja (0 V) la línea de datos 0 para terminar el impulso.

Si ANGULO=0, el impulso durará un período de un milisegundo; si ANGULO=128 (en el BBC Micro y el Commodore 64) durará 1,5 milisegundos y la palanca se moverá hasta un punto medio.

Un impulso, no obstante, no es suficiente para mantener la posición de un servomotor. Éste debe recibir una corriente de impulsos, refrescando al motor aproximadamente cada 20 milisegundos. Existen dos maneras de generar una corriente de impulsos:
1) Simplemente empleando un bucle de espera para hacer una pausa entre los impulsos. Esto significa que mientras se efectúa el bucle el ordenador no puede hacer ninguna otra cosa.
2) Utilizando "interrupciones", que permiten que el ordenador ejecute otro programa (por lo general en BASIC) casi simultáneamente. Este programa de segundo término puede indicar a los motores hacia dónde moverse.

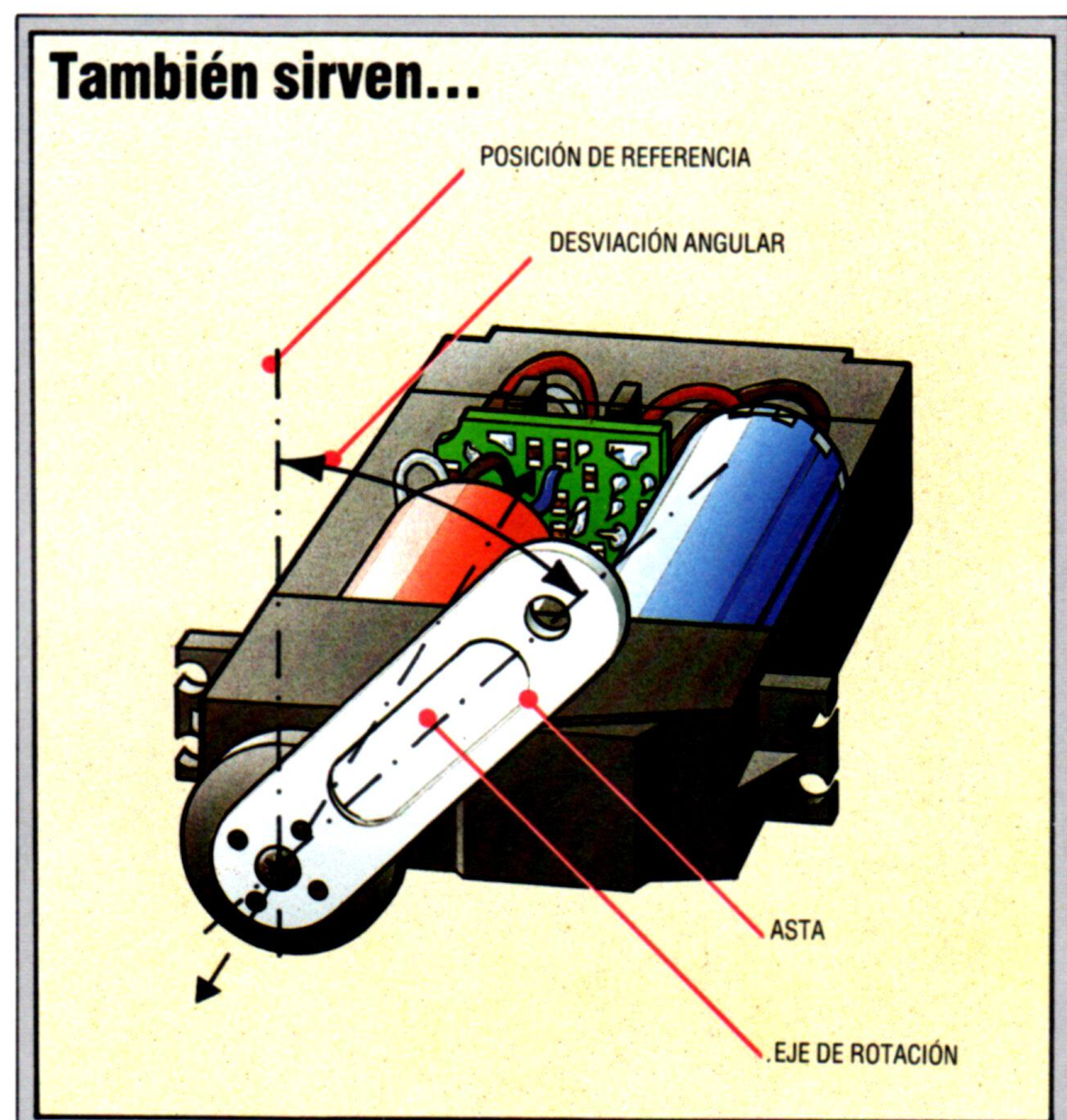
Tanto el BBC Micro como el Commodore 64 utilizan procesadores de la serie 6500, que poseen dos patillas para interrupciones: NMI e IRQ. La segunda, la línea de interrupción, la emplearemos para nuestras tareas de temporización. Cada vez que en la línea IRQ aparece un impulso, el procesador interrumpe lo que esté haciendo y comienza a ejecutar el programa de tratamiento de interrupciones. Una vez completado, retorna (RTI) al mismo punto en que se encontraba cuando fue interrumpido.

Tanto el BBC Micro como el Commodore 64 utilizan un sistema de interrupciones para sus sistemas operativos. El BBC Micro genera 100 interrupciones por segundo (una cada 10 milisegundos) y el 64 posee un coeficiente de 60 por segundo. A cada interrupción se actualizan los temporizadores del sistema, se explora el teclado, etc. De este modo, los sistemas operativos de estas máquinas poseen relojes que generan interrupciones y poseen, asimismo, manejadores (*handlers*) para interceptarlas y usarlas.

En los dos ordenadores, las interrupciones del sistema se pueden utilizar para ejecutar el programa generador de impulsos. Las interrupciones del Commodore 64 deben interceptarse cambiando el vector de interrupción. Este vector (una dirección de dos bytes retenida en dos celdas consecutivas) le dice al procesador la posición de la rutina que trata las interrupciones. Cambiando esta dirección al punto de la rutina de impulsos, y dirigiendo al procesador otra vez hasta el manejador de interrupciones del sistema, al final del impulso, el procesador generará un impulso cada vez que se produzca una interrupción: es decir, 60 veces por segundo.

Aquí le ofrecemos las versiones en BASIC y assembly de un programa para controlar un solo servomotor en el Commodore 64.

También sirven...



La desviación del asta fijada al husillo del servomotor está determinada por la duración del impulso enviado por el ordenador controlador; cuanto más largo es el impulso, mayor es la desviación. Si el impulso se repite a intervalos de unos 20 milisegundos, el motor mantendrá el asta en una determinada posición: este "bloqueo de posición" hace que el servomotor resulte ideal para aplicaciones de control electromecánico. La mayoría de los microordenadores generan interrupciones cada 10 o 20 milisegundos, de modo que esta señal de "refresco" se puede enviar sin referencia a un programa de control en BASIC, parcheando algo de código máquina en el sistema operativo a través del vector de interrupción.
Otra ventaja de los servomotores es que sólo se necesita una línea de datos para enviar los impulsos de control y refresco, de modo que a cada servomotor se le otorga un solo bit de la puerta de datos del ordenador controlador

Control de un solo servomotor mediante el Commodore 64

La primera parte del listado fuente para el control de un solo servomotor mediante el Commodore 64 muestra cómo se alteran los vectores de interrupción (posiciones 788 y 789). Esto no se puede hacer desde BASIC, puesto que durante esta alteración se podría producir una interrupción, lo que haría que el sistema se colgara. Observe que las interrupciones se inhiben (SEI) mientras se produce la alteración y se reactivan utilizando CLI. El resto del código es la rutina para tratamiento de interrupciones para controlar un servomotor.
El programa de llamada en BASIC muestra todo lo necesario para cargar las rutinas en código máquina, prepara la puerta para el usuario y después coloca (POKE) valores en la posición de memoria $3000 (12288) según la tecla (de 1 a 9) que se pulse. Un motor conectado a la puerta para el usuario debería entonces de moverse hasta una posición proporcional al valor de la tecla. Pulsando E se da por concluida la sesión. Si usted posee un ensamblador, entre el listado fuente y ensámblelo en un archivo objeto que subsiguientemente se pueda cargar mediante el programa de llamada en BASIC. De no ser así, entre el cargador en BASIC para el código máquina y ejecútelo para cargar el código en la memoria. Digite NEW antes de cargar y ejecutar el programa. Si utiliza el cargador en BASIC, puede omitir las líneas 30 y 40.
Nota: Si hay algo mal en un programa que utiliza interrupciones, es muy fácil que todo el sistema se corrompa. Por tanto, conviene guardar el programa antes de ejecutarlo

Código fuente

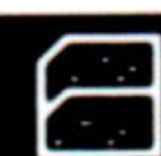
```
;.........................................
;.........................................
;..                                     ..
;..       MANEJADOR DE UN               ..
;.. .........  SERVO CBM                ..
;.........................................
;.........................................
;
PUERTA=56579   ; REG DATOS PUERTA USUARIO
ANGULO=12288   ; POSICION VALOR ANGULO
;
*=$0334
;
        SEI             ;APAGAR INTERRUPCIONES
        LDA $0314       ;VECTOR IRQ EXISTENTE
        LDX $03C4
        STA $03C4
        STA $0314
        LDA $0315
        LDX $03C5
        STA $03C5
        STX $0315
;
        CLI             ;VOLVER A ENCENDER
        RTS             INTERRUPCIONES
;
;... MANEJADOR DE EVENTOS ...
;
        PHP
        PHA
        TYA             ;SALVAR REGISTROS
        PHA             ;EN LA PILA
        TXA
        PHA
        LDA #$FF
        STA PUERTA
        LDY #$FF
BUCLE
        DEY             ;BUCLE DEMORA
        BNE BUCLE       ;APROX 1MSEG
;
        LDY ANGULO
BUCLE1
        DEY             ;DESCUENTA IMPULSO
        BNE BUCLE1
;
        LDA #$00
        STA PUERTA      ;REGISTRO DATOS CERO
;
        PLA
        TAX             ;RESTAURAR REGISTRO
        PLA             ;VALORES
        TAY
        PLA
        PLP
;
        JMP $EA31
```

Programa cargador en BASIC

```
10 REM **** CARGADOR EN BASIC ****
20 REM **** PARA UN SOLO SERVO ****
30 :
40 FOR I=820 TO 882
50 READ A:POKE I,A
60 CC=CC+A
70 NEXT I
80 READ CS:IF CC<>CS THEN PRINT
   "ERROR EN SUMA DE CONTROL":STOP
100 DATA120,173,20,3,174,196,3,141,196
110 DATA3,141,20,3,173,21,3,174,197,3
120 DATA141,197,3,142,21,3,88,96,8,72
130 DATA152,72,138,72,169,255,141,3
140 DATA221,160,255,136,208,253,172,0
150 DATA48,136,208,253,169,0,141,3,221
160 DATA104,170,104,168,104,40,76,49
170 DATA234
180 DATA7170:REM*SUMA DE CONTROL*
```

Programa de llamada en BASIC

```
10 REM **** UN SOLO SERVOMOTOR ****
20 :
30 DN=8:REM SI CASSETTE ENTONCES
   DN=1
40 IF A=0 THEN A=1:LOAD
   "SINGSERV.HEX",8,1
50 POKE 964,79:POKE 965,3:REM APUNTAR
   A MANEJADOR IRQ
60 RDD=56577:POKE RDD,255:REM TODAS
   SALIDA
70 MC=820:SYS MC:REM ESTABLECER
   VECTOR IRQ
80 POKE 53265,PEEK(53265)AND239:REM
   BORRAR PANTALLA
90 :
100 GET K$:IF K$=" " THEN100:REM
    ESPERAR PULSACION TECLA
110 REM ** ALTERAR POSICION MOTOR **
120 IF ASC(K$)>48 AND ASC(K$)<58 THEN
    POKE 12288,VAL(K$)*20
130 IF K$<>"E" THEN 80:REM "E" PARA
    SALIR
140 END
```



Formas simétricas

El LOGO es un lenguaje particularmente útil para la investigación de patrones y simetrías

Existen cuatro clases de transformación que podemos aplicar a una figura bidimensional sin que su forma sufra ninguna modificación (aunque podría variar su posición). Estas transformaciones son: traslación, rotación, reflexión y reflexión con deslizamiento. Nuestro diagrama refleja cómo cambia la posición de una forma en función de cada una de estas transformaciones.

Se dice que una figura es *simétrica* si podemos transformarla en una o más de estas formas y dejar inalteradas tanto su posición como su forma. Las formas finitas (como los polígonos y las letras del alfabeto) deben tener simetrías basadas en la reflexión y la rotación, ya que las traslaciones y las reflexiones de deslizamiento cambiarán sus posiciones.

Para investigar estas simetrías resulta útil tener procedimientos en LOGO para hacer reflejar y rotar formas. Comenzaremos por analizar la tarea de reflejar una forma en una línea que pase por el origen y posea una orientación dada.

Es más sencillo si damos por sentado que el procedimiento para dibujar la forma es de estado transparente (es decir, deja a la tortuga en la misma posición y con la misma orientación que tenía antes de que se ejecutara el procedimiento). Nuestra tarea se divide entonces en dos partes: primero, hemos de hallar las coordenadas y la orientación del punto de partida de la reflexión que corresponda al punto de partida de la forma original. Es necesario llevar a cabo la segunda tarea antes de empezar a dibujar la forma. La misma supone cambiar todos los giros hacia la derecha del procedimiento para dibujar la forma por giros hacia la izquierda, y todos los giros hacia la izquierda por giros hacia la derecha. Una manera de hacerlo consiste en reemplazar todas las RT y LT del procedimiento mediante un procedimiento llamado GIRO, definido de la siguiente manera:

```
TO GIRO :A
  RT :DIR* :A
END
```

De modo que ahora podemos definir un cuadrado como:

```
REPEAT 4 [FD 50 GIRO 90]
```

Para utilizar este procedimiento debemos primero poner a 1 la variable local DIR. Por lo tanto, MAKE "DIR 1 CUADRADO dibujará un cuadrado. Para reflejar el cuadrado en el eje, todo lo que hay que hacer es digitar MAKE "DIR (–1) y luego CUADRADO. Pruébelo y vea lo que sucede.

El procedimiento para posicionar la tortuga antes de dibujar la reflexión depende en cierta medida de la trigonometría:

```
TO REFLEJAR :A
  MAKE "O ORIENTACION
  MAKE "XANT XCOR
  MAKE "ANGULO (ATAN :YANT :XANT)–90+:A
  MAKE "R SQRT (:XANT*:XANT+:YANT*:YANT)
  PU
  SETXY 0 0
  SETH :A + :ANGULO
  FD :R
  SETH 2* :A+:H
  PD
  MAKE "DIR :DIR*(–1)
END
```

Ahora este procedimiento se puede utilizar para ver el efecto de reflexiones en varias líneas a través del origen. Pruebe con:

```
MAKE "DIR 1
PU SETXY 40 70 PD
CUADRADO
REFLEJAR 60
CUADRADO
```

Si la forma reflejada se halla completamente en la parte superior del original, se dice que posee una "simetría reflexiva", respecto a esa línea. Pruebe con:

```
MAKE "DIR 1
PU SETXY 0 0 PD
CUADRADO
REFLEJAR 45
CUADRADO
```

Se podría escribir un procedimiento similar para hacer girar una forma respecto a un punto dado a través de un ángulo dado, pero eso lo dejaremos para que lo escriba usted mismo.

Algunos patrones, como los de los papeles pintados, utilizan en su diseño la misma forma repetidamente. Se pueden efectuar traslaciones y reflexiones con desplazamiento que muevan el patrón entero y, aun así, lo dejen exactamente tal como estaba. Por el momento nos concentraremos en patrones que impliquen traslaciones a lo largo de una única línea, dejando los patrones bidimensionales para el próximo capítulo.

Las combinaciones de las cuatro transformaciones fundamentales dan origen a apenas siete tipos de patrones sobre una línea recta. Todas estas posibilidades se muestran en nuestro segundo diagrama en forma de un simple motivo "PATA". Hemos construido procedimientos para dibujar los siete patrones a partir de cualquier MOTIVO empleando los procedimientos MOVER para traslación, GIRAR para rotación, e I.MOTIVO, que convierte todos los giros RT de MOTIVO en giros LT, y todos los giros LT en RT.

Hemos utilizado las facilidades para proceso de listas del LOGO para escribir el procedimiento I.MOTIVO reescribiendo MOTIVO. El procedimiento que empleamos para hacer esto es:

```
TO REESCRIBIR :PROC
   OUTPUT REESCRIBIR.PROC TEXT :PROC
END
```

REESCRIBIR toma el texto de un procedimiento especificado, lo modifica y lo crea bajo otro nombre. Da por sentado que el procedimiento sobre el cual está trabajando está escrito con primitivas del LOGO y no contiene ningún subprocedimiento. REESCRIBIR contiene una llamada a los siguientes procedimientos:

```
TO REESCRIBIR.PROC :TEXTO
   IF :TEXTO=[ ] THEN OUTPUT [ ]
   OUTPUT FPUT REESCRIBIR.LINEA FIRST :TEXTO
   REESCRIBIR.PROC BUTFIRST :TEXTO
END
```

Este procedimiento divide la tarea de reescribir el procedimiento de entrada en líneas individuales, mediante la llamada al siguiente procedimiento:

```
TO REESCRIBIR.LINEA :LINEA
   IF :LINEA=[ ] THEN OUTPUT [ ]
   IF LIST? FIRST :LINEA THEN OUTPUT FPUT
   REESCRIBIR.LINEA FIRST :LINEA REESCRIBIR.
   LINEA
   BUTFIRST :LINEA
   OUTPUT FPUT
   CAMBIAR.PALABRA FIRST :LINEA
   REESCRIBIR.LINEA BUTFIRST :LINEA
END
```

REESCRIBIR.LINEA realiza el proceso de cada línea, pasándole las palabras individuales a CAMBIAR.PALABRA para que ésta se encargue de ellas. La línea que comienza con IF LIST? es necesaria para tratar con una situación en la que MOTIVO contenga una sentencia REPEAT. Si usted no utiliza esta sentencia en sus procedimientos MOTIVO, entonces sí puede eliminar la línea de este procedimiento. El listado para CAMBIAR.PALABRA es:

```
TO CAMBIAR.PALABRA :PALABRA
   IF (ANYOF :PALABRA="RT :PALABRA=
   "RIGHT) THEN OUTPUT "LEFT
   IF (ANYOF :PALABRA="LT :PALABRA=
   "LEFT) THEN OUTPUT "RIGHT
   OUTPUT :PALABRA
END
```

Este procedimiento verifica cada palabra individual y realiza las modificaciones necesarias. Habiendo entrado todos estos procedimientos, veamos ahora cómo funcionan. En primer lugar, es preciso definir una forma simple, como por ejemplo:

```
TO TRI
   REPEAT 3 [FD 50 RT 120]
END
```

Ahora entre DEFINE "REF REESCRIBIR "TRI, y llame a REF. Su definición ha de ser:

```
TO REF
   REPEAT 3 [FD 50 LEFT 120]
END
```

Es bastante factible escribir un procedimiento REESCRIBIR más general que también reescriba cualquier procedimiento llamado por el procedimiento principal. Si intenta escribirlo, ¡tenga cuidado con los procedimientos recursivos! También necesitará poder comprobar si una palabra es el nombre de un procedimiento.

Transformaciones isométricas
Las transformaciones que alteran la posición de un objeto pero no su forma se llaman isometrías. Existen cuatro tipos básicos de transformaciones isométricas: traslación, rotación, reflexión y reflexión con deslizamiento. La traslación es un simple "deslizamiento" de la figura original. La rotación hace girar la forma alrededor de algún punto central especificado. La reflexión implica el movimiento de puntos a través de una línea de espejo, de modo que cada punto de la forma final esté a la misma distancia hacia un lado de la línea que el punto correspondiente del original hacia el otro lado. La reflexión con deslizamiento es una reflexión y una traslación. Mientras que la traslación y la rotación conservan el "sentido", la reflexión y la reflexión con deslizamiento lo cambian: imagínese, por ejemplo, una palabra reflejada en un espejo

Ian McKinnell

Los siete patrones de franjas

Sería posible (y elegante desde el punto de vista matemático) construir los patrones a partir de procedimientos para las cuatro transformaciones básicas. Los procedimientos para dibujar patrones hacen uso de estos tres procedimientos auxiliares:

```
TO POSICION
   HT
   PU
   SETXY – 125 0
   PD
END
```

Éste posiciona la tortuga en el lado izquierdo de la pantalla, lista para empezar a dibujar.

```
TO MOVER
   PU
   RT 90
   FD 50
   LT 90
   PD
END
```

MOVER realiza la traslación requerida.

```
TO GIRAR
   RT 180
END
```

GIRAR efectúa la rotación que necesitamos.

Para utilizar estos procedimientos defina primero un procedimiento de forma (digamos, FORMA) que sea de estado transparente y que no incluya llamadas a subprocedimientos. Después ya puede dibujar el primer patrón empleando FORMA como su motivo, entrando PATRON1 "FORMA.



Siete patrones de franjas

Para ejecutar los procedimientos de patrones debe tener en el espacio de trabajo los siguientes procedimientos: REESCRIBIR.PROC, REESCRIBIR.LINEA, CAMBIAR.PALABRA, POSICION, MOVER y GIRAR. Los siete patrones posibles son:

```
TO PATRON1 :PROC
  DEFINE "MOTIVO TEXT :PROC
  POSICION
  REPEAT 6 [MOTIVO MOVER]
END

TO PATRON2 :PROC
  DEFINE "MOTIVO TEXT :PROC
  DEFINE "I.MOTIVO REESCRIBIR.PROC
  TEXT :PROC
  POSICION
  REPEAT 3 [MOTIVO MOVER.GIRAR
  I.MOTIVO GIRAR MOVER]
END

TO PATRON3 :PROC
  DEFINE "MOTIVO TEXT :PROC
  DEFINE "I.MOTIVO REESCRIBIR.PROC
  TEXT :PROC
  POSICION
  REPEAT 6 [MOTIVO I.MOTIVO MOVER]
END

TO PATRON4 :PROC
  DEFINE "MOTIVO TEXT :PROC
  DEFINE "I.MOTIVO REESCRIBIR.PROC
  TEXT :PROC
  POSICION
  REPEAT 6 [MOTIVO GIRAR MOTIVO
  GIRAR MOVER]
END

TO PATRON5 :PROC
  DEFINE "MOTIVO TEXT :PROC
  DEFINE "I.MOTIVO REESCRIBIR.PROC
  TEXT :PROC
  POSICION
  REPEAT 3 [MOTIVO I.MOTIVO MOVER
  GIRAR MOTIVO I.MOTIVO GIRAR
  MOVER]
END

TO PATRON6 :PROC
  DEFINE "MOTIVO TEXT :PROC
  DEFINE "I.MOTIVO REESCRIBIR.PROC
  TEXT :PROC
  POSICION
  REPEAT 6 [MOTIVO GIRAR I.MOTIVO
  GIRAR MOVER]
END

TO PATRON7 :PROC
  DEFINE "MOTIVO TEXT :PROC
  DEFINE "I.MOTIVO REESCRIBIR.PROC
  TEXT :PROC
  POSICION
  REPEAT 6 [MOTIVO I.MOTIVO GIRAR
  MOTIVO I.MOTIVO GIRAR MOVER]
END
```

Es necesario ejecutar estos procedimientos con algún procedimiento para dibujar un motivo adecuado. El motivo que hemos utilizado es:

```
TO PATA
  FD 50
  RT 90
  FD 20
  BK 20
  LT 90
  BK 50
END
```



Un motivo alternativo es:

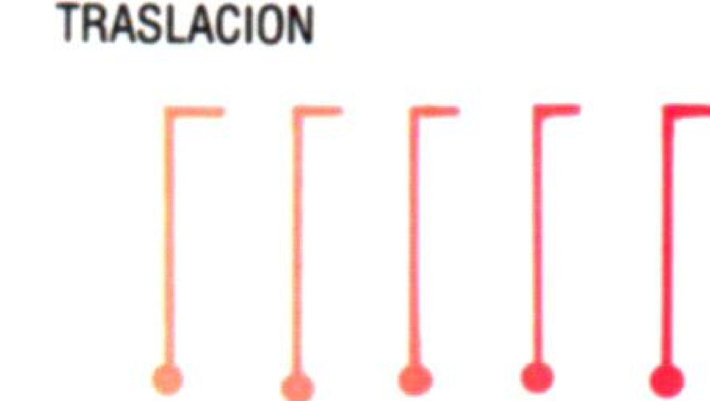
```
TO FIG
  RT 30
  FD 20
  LT 50
  FD 20
  RT 90
  FD 10
  REPEAT 4 [FD 20 RT 90]
  BK 10
  LT 90
  BK 20
  RT 50
  BK 20
  LT 30
END
```

Complementos al LOGO

ATAN y TOWARDS no existen en el LOGO Atari, ni tampoco existe un recambio sencillo. Ello incide en los procedimientos REFLEXION y ROTACION, pero no en los PATRON.
El LOGO Atari no posee TEXT ni DEFINE como primitivas, si bien el manual ofrece un método para definirlas. Usted puede escribir I.MOTIVO modificando MOTIVO mediante el editor

Logomotivo

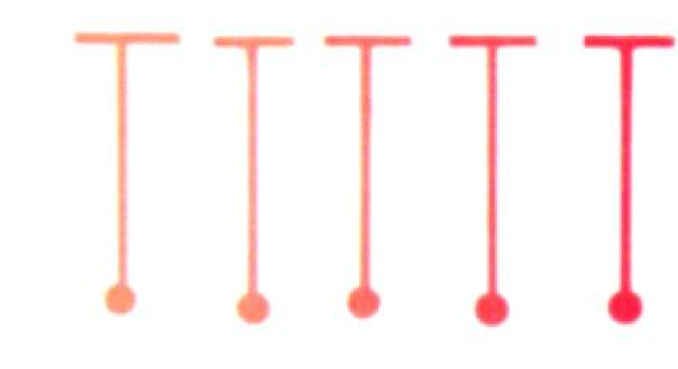
TRASLACION



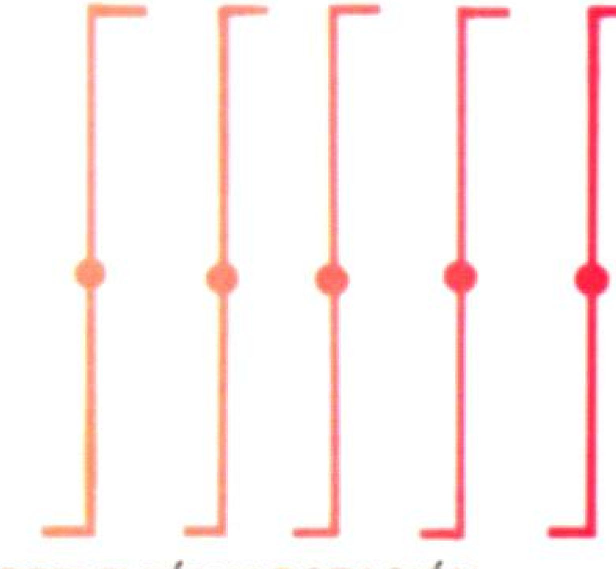
REFLEXIÓN CON DESLIZAMIENTO



DOS REFLEXIONES



TRASLACIÓN Y ROTACIÓN



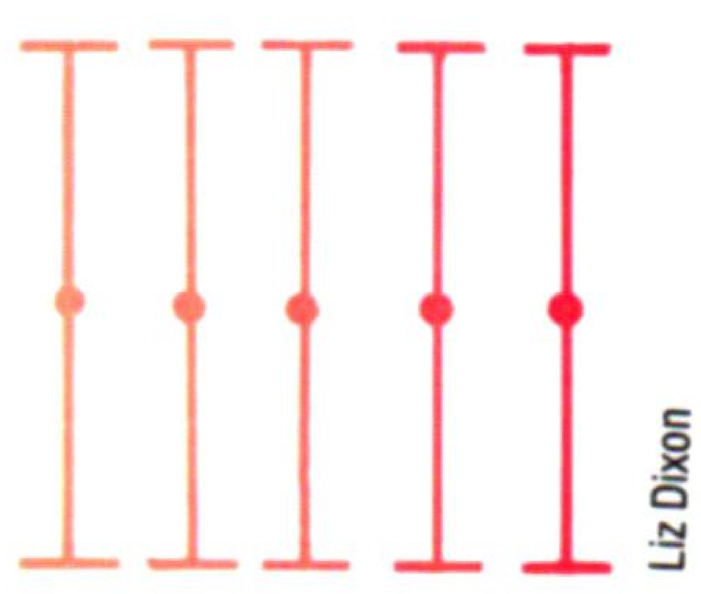
REFLEXIÓN Y ROTACIÓN



TRASLACIÓN Y REFLEXIÓN



TRASLACIÓN Y DOS REFLEXIONES



Siete de la misma clase
Las cuatro transformaciones isométricas primitivas se pueden combinar de diversas formas para producir siete patrones exclusivos, tal como vemos en la ilustración. En cada caso partimos del motivo "pata" y todas las traslaciones se efectúan en la dirección del eje *x*

Liz Dixon

Jaque al chip

En el Commodore 64 es fácil reubicar posiciones de memoria. Lo demostraremos con una rutina para diseñar y almacenar un máximo de ocho visualizaciones

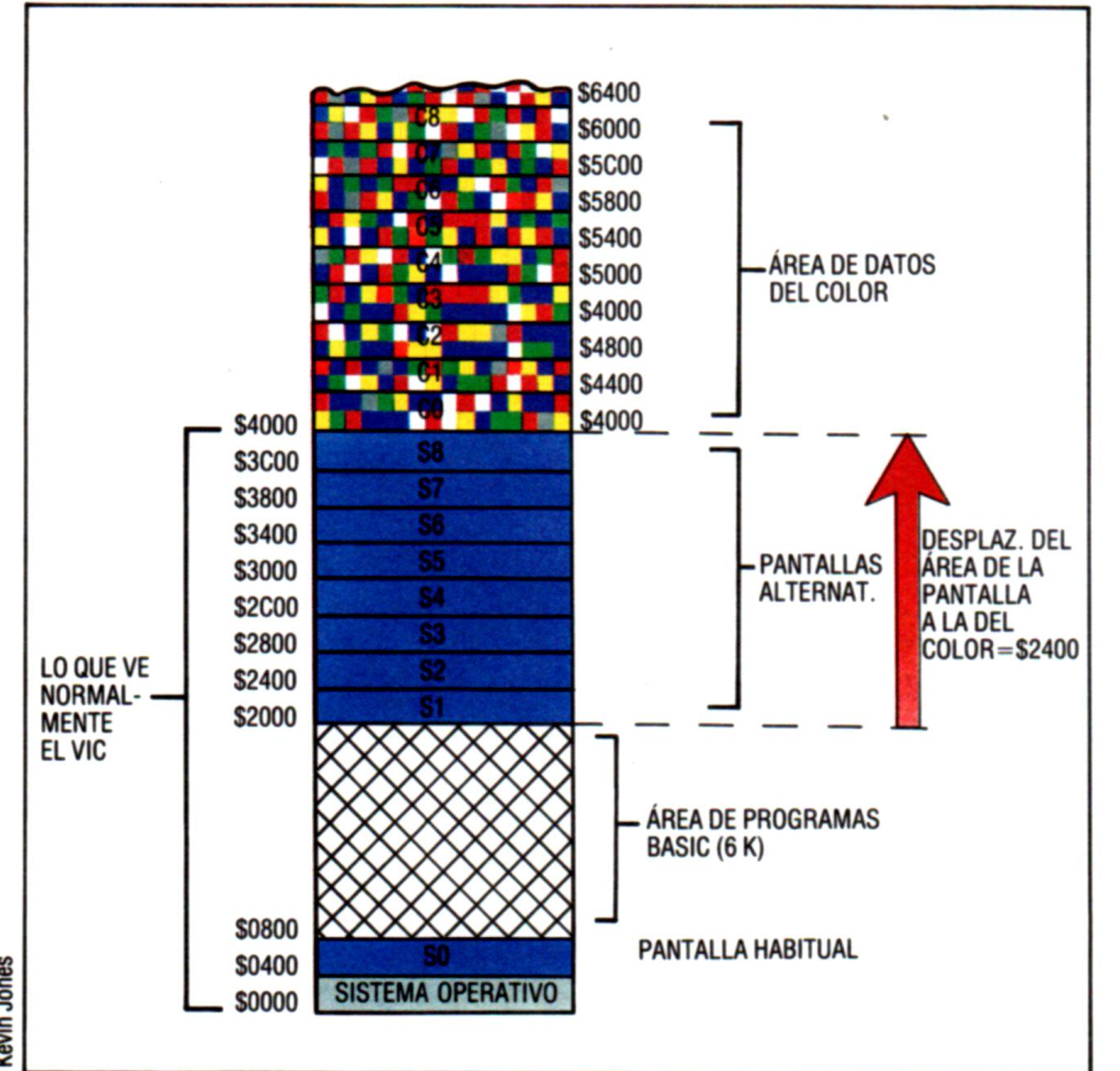
Panorama VIC
El chip controlador de video (VIC) del Commodore 64 puede "ver" 16 K de memoria. Generalmente se trata de los primeros 16 K, del $0000 al $3FFF, pero puede hacerse que "mire" hacia cualquier otro de los tres bloques de 16 K, alterando el contenido de uno de los registros de control del VIC. El programa "Pantallas alternativas" establece hasta nueve mapas de pantalla dentro del área normal de 16 K que puede ver el chip. Los mapas correspondientes a cada pantalla se encuentran en el área de la RAM que está justo encima del área vista por el VIC, disponiendo cada pantalla, salvo la pantalla 0, de un desplazamiento constante de $2400 respecto a su mapa de color

En el Commodore 64, un chip especial controla la visualización y el tratamiento de sprites. Es el chip denominado VIC II. El chip VIC accede a diversos sectores de la memoria para obtener información útil para crear la visualización que se nos presenta en la pantalla. Estas áreas incluyen la ROM de caracteres, que contiene la forma del trazado de los caracteres; la RAM del color, que guarda la información del color en pantalla, y la RAM de la pantalla. Esta última contiene información de los caracteres que pueden visualizarse en las 1 000 posiciones (25 filas por 40 columnas) que constituyen el espacio de la pantalla.

En el momento de conectar el Commodore 64, el chip VIC supone que la pantalla está posicionada en esos 1 000 bytes que comienzan en la posición 1024 ($0400) y accede a esta área para buscar la información inicial de la pantalla. Pero si se altera el valor de un registro dentro del chip VIC es posible redireccionar el citado chip a otra zona de memoria, normalmente en los primeros 16 K de la memoria. Los cuatro bits superiores del registro de control en el VIC se encuentran en la posición 53272 ($D018) y determinan el bloque de K que ha de ser tomado como la pantalla, dentro de la citada área de 16 K. He aquí un cuadro que ilustra los valores de los bits que corresponden a cada una de las 16 posibles posiciones de pantalla:

Patrón de bits	Inicio pantalla	
0000XXXX	0	$0000
0001XXXX	1024	$0400 *
0010XXXX	2048	$0800
0011XXXX	3072	$0C00
0100XXXX	4096	$1000
0101XXXX	5120	$1400
0110XXXX	6144	$1800
0111XXXX	7168	$1C00
1000XXXX	8192	$2000
1001XXXX	9216	$2400
1010XXXX	10240	$2800
1011XXXX	11264	$2C00
1100XXXX	12288	$3000
1101XXXX	13312	$3400
1110XXXX	14336	$3800
1111XXXX	15360	$3C00

* = Posición por defecto



Kevin Jones

Para que el chip VIC traslade la pantalla a otra área, hemos de cambiar los cuatro bits superiores de la posición 53272 ($D018) por los valores indicados en este cuadro. Los cuatro bits inferiores no deben ser modificados (los hemos indicado con XXXX), pues controlan otra función. Para poner a cero los cuatro bits superiores sin cambiar el valor de los inferiores, debemos operar el contenido del registro con un AND lógico empleando el número 15 (00001111 en binario). Una vez hecho esto operaremos con OR el nuevo contenido del registro empleando el valor que deseamos. Para colocar la pantalla en la última área que el chip puede ver (es decir, la que comienza en 15360, o sea, $3C00), habremos de emplear un OR con el contenido del registro y el número 240 (binario, 11110000). En BASIC se encarga de todo esto la siguiente instrucción POKE:

```
POKE 53272,(PEEK(53272)AND15)OR240
```

Antes de poder escribir algo en nuestra nueva pantalla, tendremos también que decirle al sistema operativo del Commodore 64 que la posición de la pantalla fue alterada. Lo que conseguimos colocando el byte *hi* de la nueva dirección de inicio de la pantalla en la posición 648 ($0288). Para la pantalla más alta resulta ser $3C, y se obtiene fácilmente en BASIC dividiendo por 256 la dirección de inicio de pantalla.

Una vez cambiado el contenido de estos dos re-

gistros podemos hacer uso de la nueva pantalla como si nada hubiera ocurrido. Observe que si hubiera que cambiar la posición de la pantalla en BASIC habría que escribir un pequeño programa para hacerlo.

Es posible hacer uso de la facilidad con que en un Commodore 64 puede moverse la pantalla para obtener varias utilidades de interés. En concreto, podemos cambiar la visualización con rapidez y sin dificultad. Sólo que si deseamos mover la pantalla hay que mover también la RAM correspondiente del color, ya que la visualización quedará sin gracia, faltándole los datos adecuados de la RAM del color. A pesar de que podemos establecer varias áreas dentro de la memoria y cambiarlas para pantalla, sólo se dispone de una inamovible área RAM del color, lo que significa que, para retener varias visualizaciones individuales de pantalla, debemos reservar zonas de memoria capaces de contener los 1 000 bytes de datos de color para cada pantalla. Cuando se desea visualizar una pantalla hay que copiar la información en la RAM del color y guardar los datos en una de las áreas de la memoria escogidas para albergar la RAM del color, antes de cambiar a otra pantalla diferente.

Organización de la memoria

La tarea principal de esta utilidad es organizar la memoria para pantallas alternas (junto con sus correspondientes áreas para datos del color), y realizar la transferencia de bloques de memoria. Dado que el chip VIC puede "ver" 16 K de memoria, podemos diseñar un sistema que nos permita disponer hasta de ocho pantallas diferentes y un programa BASIC a medida. En un diagrama adjunto hemos ilustrado la organización de la memoria que emplea la utilidad.

Para estar seguros de que no va a ser machacada ningún área de pantalla o de color a causa de algún programa BASIC, debemos bajar la frontera superior de la memoria BASIC. Esto lo hace la siguiente instrucción en nuestro programa en BASIC:

```
POKE 55,0:POKE 56,32:CLR
```

La dirección de base de cualquier pantalla se calcula a partir de su número mediante esta fórmula:

Base pantalla = $1C00 + ($0400*Número pantalla)

La dirección de base de la correspondiente área del color se calculará añadiendo un desplazamiento a la dirección de base de la pantalla. La fórmula es:

Base color = $2400 + Base pantalla

Las áreas del color pueden ubicarse en cualquier lugar de la RAM, pero es recomendable colocarlas justo encima de la última pantalla visible para el chip VIC. Observe que está incluida un área de color para la pantalla 0, que es la pantalla habitual. Para esta particular área de color, el desplazamiento ha de ser diferente y deberá ser tenido en cuenta en nuestro programa. El registro de control del VIC y el registro del sistema operativo pueden establecerse para cualquier pantalla mediante:

Registro VIC = $70 + ($10*Número pantalla)
Registro OS = Byte HI de la dirección base de pantalla

Además, para manejar el establecimiento de registros y realizar las transferencias adecuadas de los datos de color desde y hacia el área de RAM de color, el programa almacena también el color del fondo y el borde o recuadro de la pantalla. Estas dos tareas se controlan mediante un par de registros en el chip VIC: 53280 ($D020) controla el color del borde o recuadro (*border*) y 53281 ($D021) controla el color del fondo de la pantalla (*paper*). La utilidad establece una tabla dentro de su área para almacenar estos atributos para cada pantalla.

Modos operativos

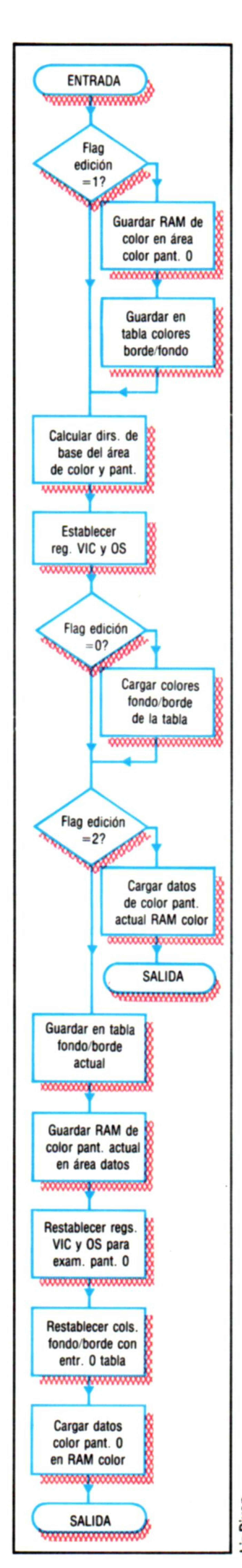
La utilidad tiene dos modos de operar: puede editar(*edit*) o visualizar (*display*) una pantalla seleccionada. En cada caso el número de pantalla a emplear debe colocarse (POKE) en 49152 ($C000). Para conseguir que sea la misma llamada SYS nos servimos de un flag especial que indicará cuál ha sido el modo escogido. Este flag se coloca (POKE) en la posición 49153 ($C001).

0: indica el modo visualización
1: indica el modo edición

El modo edición funciona de forma poco habitual, para poder emplear todas las facilidades del editor de pantalla (como cambio de color de texto, establecimiento de modo invertido y borrado de pantalla). La utilidad debe ser llamada antes y poner a 1 el flag de edición. Después guardará el área de color normal de pantalla y los colores de papel y recuadro, pondrá el valor adecuado en los registros del sistema operativo y del VIC. En este punto, el programa en BASIC de llamada toma el control poniendo el cursor en su inicio y empleando la instrucción INPUT del BASIC. Esta instrucción esperará un carácter de retorno (13 en ASCII) antes de proseguir. Mientras, todas las funciones del editor de pantalla pueden ser usadas de la manera habitual para editar la pantalla seleccionada; cuando la tarea esté concluida se pulsará Return.

El programa BASIC debe posteriormente llamar por segunda vez a la utilidad, pero en esta ocasión el flag de edición se pone a 2. Esto hace que se guarden los datos del color y los colores del fondo-/borde de la pantalla con los que se ha trabajado antes de restablecer la pantalla habitual. Si se desea cambiar el color del borde o del fondo, estos colores deben colocarse (POKE) en las posiciones 49154 ($C002) y 49155 ($C003) respectivamente. Aunque la rutina está diseñada para esta tarea concreta, incorpora además rutinas generales para establecer registros de control y copiar los datos desde o hacia la RAM del color. No le será, por tanto, difícil idear una utilidad para sus propias especificaciones partiendo de las rutinas generales.

El programa de llamada en BASIC está diseñado para visualizar un menú que dé la posibilidad de editar o visualizar. La rutina de visualización llama a cada una de las ocho pantallas diferentes secuencialmente y respondiendo a pulsaciones del teclado. Las pantallas seguirán rotando hasta que se pulse la tecla Return. Entonces se restaurará la pantalla normal y se volverá al menú. Si se escoge la opción de edición, el usuario puede establecer los colores del fondo y del recuadro para la pantalla escogida, y puede servirse del editor de pantalla de la manera habitual para modificar el contenido de ésta. Una vez concluido, Return hace que el cuadro se almacene y se restablezca la pantalla normal.



Liz Dixon

Cargador de BASIC

```
10 REM **************************************
20 REM **        CARGADOR DE BASIC PARA      **
30 REM **        PANTALLAS ALTERNATIVAS      **
40 REM **************************************
50 :
60 FORI=49152 TO 49439
65 READA:POKE I,A
70 CC=CC+A
75 NEXT
80 READ CS:IFCS<>CC THENPRINT"CHECKSUM ERROR":STOP
100 DATA255,255,0,0,191,255,0,0,255
110 DATA191,0,0,191,191,0,0,255,191,0
120 DATA0,255,191,0,0,255,255,0,173,1
130 DATA192,201,1,208,30,169,0,141,6
140 DATA192,169,64,141,7,192,32,224
150 DATA192,162,0,32,246,192,173,2,192
160 DATA141,32,208,173,3,192,141,33
170 DATA208,173,0,192,208,6,32,32,193
180 DATA76,139,192,169,0,141,4,192,141
190 DATA5,192,174,0,192,173,5,192,24
200 DATA105,4,202,208,250,24,105,28
210 DATA141,5,192,24,105,36,141,7,192
220 DATA173,0,192,162,4,10,202,208,252
230 DATA24,105,112,141,8,192,173,24
240 DATA208,41,15,13,8,192,141,24,208
250 DATA173,5,192,141,136,2,173,1,192
260 DATA208,6,174,0,192,32,211,192,201
270 DATA2,240,4,32,189,192,96,174,0
280 DATA192,32,246,192,32,224,192,32
290 DATA32,193,169,0,141,6,192,169,64
300 DATA141,7,192,162,0,32,211,192,32
310 DATA189,192,96,173,6,192,133,251
320 DATA173,7,192,133,252,169,0,133
330 DATA253,169,216,133,254,32,3,193
340 DATA96,189,9,192,141,32,208,189,18
350 DATA192,141,33,208,96,173,6,192
360 DATA133,253,173,7,192,133,254,169
370 DATA0,133,251,169,216,133,252,32,3
380 DATA193,96,173,32,208,157,9,192
390 DATA173,33,208,157,18,192,96,162,3
400 DATA160,0,177,251,145,253,136,208
410 DATA249,230,252,230,254,202,48,10
420 DATA208,240,177,251,145,253,160
430 DATA231,208,232,96
440 DATA36606;REM*CHECKSUM*
```

Pantallas alternativas

```
3 REM ********************************************
5 REM **                                        **
10 REM **          PANTALLAS ALTERNATIVAS        **
11 REM **                                        **
12 REM ********************************************
13 :
15 DN=8:REM PARA CASSETTE DN=1
20 IF A=0 THEN A=1:LOAD "ALT SCREENS.HEX",DN,1
25 POKE55,0:POKE56,32:CLR:REM BAJA FRONTERA MEMORIA
40 SCNUMB=49152: REM NUMERO PANTALLA (SCREEN)
70 EDITFG=49153: REM 0=VISUALIZA PANTALLA
75 :             REM 1=EDITA PANTALLA
77 :             REM 2=COPIA RAM COLOR
80 ALT=49179:    REM DIRECCION INICIO EN COD MAQ
85 BRDCOL=49154REM COLOR BORDE
87 PAPCOL=49155REM COLOR FONDO (PAPER)
90 :
95 REM **** MENU PRINCIPAL ****
96 :
100 PRINTCHR$(147):REM LIMPIA PANTALLA
105 PRINTCHR$(154):REM LETRAS AZULES LT
110 DN$="      ":REM Q=CARACT CURSOR ABAJO
130 PRINTTAB(8);DN$;"CBM64 PANTALLAS ALTERN"
135 PRINTTAB(8);"————————————————————————"
140 PRINTTAB(5);DN$;"F1 — EDITA DIBUJO"
160 PRINTTAB(5);DN$;"F3 — VISUALIZA SECUENCIA DIBUJO"
200 :
210 GETA$:IFA$=" " THEN 210:REM ESPERA PULSACION TECLA
220 IFA$="   " THEN GOSUB 1000
230 IFA$="   " THEN GOSUB 1500
260 GOTO 100
999 :
1000 REM **** EDITA PANTALLA ****
1002 :
1005 EF=1:REM ACTIVA MODO EDICION
1010 PRINTCHR$(147)
1020 PRINTTAB(16);DN$;"MODO EDICION"
1030 PRINTDN$;INPUT"NUM.PANTALLA";SN$
1050 IFASC(SN$)<48 OR ASC(SN$)>56 THEN 1030
1060 PRINT DN$; :INPUT "COLOR BORDES";BC$
1070 IFVAL(BC$)=0 AND BC$<>"0" THEN 1060
1080 PRINT DN$; :INPUT "COLOR FONDO";PC$
1090 IF VAL(PC$)=0 AND PC$<>"0" THEN 1080
1100 :
1110 POKEEDITFG,EF
1120 POKESCNUMB,VAL(SN$)
1130 POKEBRDCOL,VAL(BC$)
1140 POKEPAPCOL,VAL(PC$)
1150 SYS ALT
1155 REM ** ESPERA RETURN **
1162 INPUT "   ";X$;REM S=CURSOR EN INICIO
1165 REM ** GUARDA PANTALLA **
1170 EF=2
1175 POKE EDITFG,EF
1180 SYS ALT
1185 RETURN
1190 :
1500 REM **** VISUALIZA PANTALLA ****
1510 EF=0
1520 PRINT CHR$(147)
1545 SN=1
1550 POKE EDITFG,EF
1555 POKE SCNUMB,SN
1560 SYS ALT
1570 GET X$: IFX$=" " THEN1570:REM ESPERA PULSACION TECLA
1572 IF X$=CHR$(13) THEN 1580:REM PANTALLA HABITUAL
1575 SN=SN+1:IF SN<9 THEN 1555
1577 SN=1: GOTO 1555
1580 POKESCNUMB,0:SYS ALT
1600 RETURN
```

Commodore 64

```
;+++++++++++++++++++
;+++++++++++++++++++
;++      PANTALLAS      ++
;++     ALTERNATIVAS    ++
;++    PARA EL CBM 64   ++
;+++++++++++++++++++
;+++++++++++++++++++
;
FROM    =$FB              ;PAGINA 0
TO      =$FD              ;PUNTEROS
;
CRAMLO=$00                ;INICIO RAM COLOR (CRAM)
CRAMHI=$D8
NCOLLO=$00                ;COLOR HABITUAL
NCOLHI= $40               ;DIRECCION BASE
NSCRHI= $04               ;DIR BASE PANTALLA HABITUAL
NVCPOK=$10                ;VALOR HABITUAL REG VIC
;
BLOCKS= $03               ;NUM DE BLOQUES DE 256 BYTES
EXTRA   =$E7              ;CANT EXTRA DE 1000 BYTES
;
COLOFF= $24               ;BYTE HI DESPL COLOR
SCROFF= $1C               ;BYTE HI DESPL PANTALLA
VICOFF= $70               ;DESPL REG CONTROL DEL VIC
VCMASK=$0F                ;MASCARA BYTE LO REG CONTROL VIC
VICREG= $D018             ;REG CONTROL POSICION PANT
EDREG=$0288               ;REG NUCLEO EDICION PANT
BORDER=$D020              ;REG COLOR BORDE
PAPER   =$D021            ;REG COLOR FONDO
;
*=$C000                   ;EST PUNTERO CARGA
;
SCNUMB  *=*+1             ;NUMERO PANTALLA
EDITFG  *=*+1             ;FLAG MODO EDICION
BRDCOL  *=*+1             ;COLOR BORDE
PAPCOL  *=*+1             ;COLOR FONDO
;
SCBASE  *=*+2             ;ALMAC BASE PANTALLA
CLBASE  *=*+2             ;ALMAC BASE COLOR
VPOKE   *=*+1             ;ALMAC PROVIS PARA NUMERO VIC
BRDTAB  *=*+9             ;TABLA COLORES BORDE
PAPTAB  *=*+9             ;TABLA COLORES FONDO
;
;**** GUARDA PANTALLA 0 ****
        LDA EDITFG
        CMP #$01          ;SI ES 0 O BIEN 2
        BNE CALC          ;ENTONCES NO SE GUARDA
;
        LDA #NCOLLO
        STA CLBASE
        LDA #NCOLHI
        STA CLBASE+1
        JSR SAVE          ;GUARDA CRAM EN C0
;
        LDX #$00
        JSR SAVEBP        ;GUARDA REGISTROS B/P
;
        LDA BRDCOL
        STA BORDER        ;ESTABLECE NUEVOS B/P
        LDA PAPCOL        ;COLORES
        STA PAPER
;
;**** CALCULA BASE COLOR ****
;
CALC
        LDA SCNUMB
        BNE NOZERO
        JSR RESET         ;EST REGISTROS NORMALES
        JMP TESTIFG
;
NOZERO
        LDA #$00
        STA SCBASE
        STA SCBASE+1      ;INICIO BASE PANT
;
        LDX SCNUMB        ;CARGA NUM PANTALLA
        LDA SCBASE+1      ;SOLO BYTE HI
MULT
        CLC
        ADC #$04
        DEX
        BNE MULT
;
        CLC
        ADC #SCROFF       ;SUMA DESPL PANTALLA
        STA SCBASE+
;
        CLC
        ADC #COLOFF       ;SUMA DESPL COLOR
        STA CLBASE+1
;
;**** EST REGS VIC Y EDITOR ****
        LDA SCNUMB
        LDX #$04
MORE
        ASL A
        DEX
        BNE MORE          ;MULT POR 16
;
        CLC
        ADC #VICOFF       ;SUMA DESPL
        STA VPOKE
        LDA VICREG
        AND #VCMASK
        ORA VPOKE
        STA VICREG        ;EST REG VIC
;
        LDA SCBASE+1
        STA EDREG         ;EST REG EDITOR
;
;**** COMPROBACION ESTADO FLAG EDICION ****
TESTFG
        LDA EDITFG
        BNE NOLOAD        ;SI 0 MODO VISUALIZ
        LDX SCNUMB
        JSR LOADBP        ;CARGA B/P EN REGS
;
;
NOLOAD
        CMP #$02
        BEQ CONT          ;SI 2 ENTONCES GUARDA RAM
;
        JRS LOAD          ;CARGA COLOR EN GRAM
        RTS
;
CONT
        LDX SCNUMB
        JSR SAVEBP        ;GUARDA REGS BP
        JSR SAVE          ;GUARDA CRAM EN COL
        JSR RESET         ;EST REGS NORMALES
        LDA #NCOLLO
        STA CLBASE        ;CARGA CBASE CON BASE C0
        LDA #NCOLHI
        STA CLBASE+1
        LDX #$00
        JSR LOADBP        ;RECARGA COLORES INIC BORDE/FONDO
        JSR LOAD          ;CARGA COLORES PANT NORMAL
        RTS
;
;**** S/R DE TRANSFERENCIA A LA RAM ****
LOAD
        LDA CLBASE
        STA FROM          ;CARGA PUNTEROS PAGINA 0
        LDA CLBASE+1
        STA FROM+1
;
        LDA #CRAMLO
        STA TO
        LDA #CRAMHI
        STA TO+1
;
        JSR COPY          ;COPIA AREA RAM
        RTS
;**** S/R DE CARGA COLS. BORDE/FONDO ****
LOADBP
        LDA BRDTAB,X
        STA BORDER
        LDA PAPTAB,X
        STA PAPER
        RTS
;
;**** S/R DE TRANSFERENCIA DESDE RAM ****
SAVE
        LDA CLBASE
        STA TO            ;CARGA PUNTEROS PAG 0
        LDA CLBASE+1
        STA TO+1
;
        LDA #CRAMLO
        STA FROM
        LDA #CRAMHI
        STA FROM+1
;
        JSR COPY          ;COPIA AREA RAM
        RTS
;
;**** S/R DE ALMAC COLORES BORDE/FONDO ****
SAVEBP
        LDA BORDER
        STA BRDTAB,X
        LDA PAPER
        STA PAPTAB,X
        RTS
;
;**** S/R DE COPIA 1000 BYTES ****
;
COPY
        LDX #BLOCKS
        LDY #$00
NEXT
        LDA (FROM),Y
        STA (TO),Y
        DEY
        BNE NEXT
NXBLOC
        INC FROM+1
        INC TO+1
        DEX
        BMI FINISH
        BNE NEXT
        LDA (FROM),Y
        STA (TO),Y
        LDY #EXTRA        ;BYTES EXTRA
        BNE NEXT
FINISH
        RTS
;
;**** S/R REST REGS VIC Y EDICION ****
RESET
        LDA #NCOLLO
        STA CLBASE
        LDA #NCOLHI
        STA CLBASE+1
        LDA VICREG
        AND #VCMASK
        ORA #NVCPOK
        STA VICREG        ;RESTAURA REG VIC
        LDA #NSCRHI
        STA EDREG         ;RESTAURA REG ED
        RTS
```

Editorial Delta, S.A.



10070

9 788485 822836